DOCUMENT RESUME

ED 180 825 SE 029 694

AUTHOR	Akiba, Y.; And Others
TITLE	GERTS GQ User's Manual.
INSTITUTION	California State Univ., Sacramento.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	Nov 74
GRANT	NSF-GY-11496
NOTE	123p.; Prepared by the School of Engineering; Not available in paper copy due to marginal legibility of original document
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	*Computer Programs; Computers; *Digital Computers; Engineering; Input Output; *Manuals; *Programing; Programs; *Simulation; Simulators; Statistics
IDENTIFIERS	*Graphical Evaluation and Review Technique

ABSTRACT

This user's manual for the simulation program Graphical Evaluation and Review Technique (GERT) GQ contains sections on nodes, branches, program input description and format, and program output, as well as examples. Also included is a programmer's manual which contains information on scheduling, subroutine descriptions, COMMON Variables, and programming listing. (MK)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

GY-11496

ED180825

GERTS GQ

USER'S MANUAL

Y. Akiba, N. Blom,

G. Duran, and L. Dabaghian

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Mary L. Charles
NSF
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

SCHOOL OF ENGINEERING

CSU, SACRAMENTO

SE 029 694

GERTS GQ USER'S MANUAL

by

Y. Akiba, N. Blom, G. Duran, and L. Dabaghian

November 1974

Copyright 1974 by Foundation of
California State University, Sacramento

Acknowledgement: This investigation was supported by the NSF under Grant GY-11496.

Table of Contents

Part I: User's Manual

1. GERTS GQ: Introduction 1

   1.1 Scope and Development of Simulation of "Q" Networks . 1

   1.2 Description of the Simulation Terminology 2

   1.3 Features of GERTS GQ 3

2. Nodes 4

   2.1 Event Nodes 5

      2.1.1 SOURCE Nodes 5

      2.1.2 SINK Nodes 6

      2.1.3 STATISTICS Nodes 6

      2.1.4 MARK Nodes 6

   2.2 Queue Node 7

      2.2.1 Simple Queue 8

      2.2.2 Gated Queue 9

   2.3 Deterministic and Probabilistic Nodes 12

   2.4 Removal 12

3. Branches 13

   3.1 Function of Branches and Parameters 13

   3.2 Probabilistic Time Distributions 14

   3.3 Network Modification 15

      3.3.1 An Example of Network Modification. 16

   3.4 Count Types 18

4. Program Input Description and Format 19

   4.1 Card Type 1, Identification 19

   4.2 Card Type 2, Specification 20

   4.3 Card Type 3, Description of Nodes 21

4.4 Card Type 4, Time Parameters 23

4.5 Card Type 5, Gated Queues 26

4.6 Card Type 6, Branching Activity 27

4.7 Card Type 7, Network Modification 28

4.8 Card Type 8, Trace or Activity Profile . . . 30

4.9 **Multi-data Sets** 30

5. Program Output 31

5.1 Data Input Check 31

5.1.1 Data Card Type 1 and 2 31

5.1.2 Data Card Type 3 32

5.1.3 Data Card Type 4 33

5.1.4 Data Card Type 5 33

5.1.5 Data Card Type 6 33

5.1.6 Data Card Type 7 34

5.2 Trace .34

5.3 Activity Profile 36

5.4 Statistics. 38

5.4.1 Statistic Nodes 38

5.4.2 Counter Types 39

5.4.3 Queue Nodes Statistics 40

5.5 Histogram 41

6. Illustrative Examples 43

6.1 Thief of Bagdad 43

6.2 Ticket Box Office Problem 47

6.3 Wax Mystery Museum 52

Part II: Programmer's Manual. 58

1. Scheduling 59

2. Subroutine Descriptions. 59

2.1 In Core Routines 60
2.2 Overlay 1. 60
2.3 Overlay 2. 61
3. COMMON Variables 63
4. Program Listing 68
4.1 In Core Routines 68
4.2 Overlay 1 72
4.3 Overlay 2 81

Appendix A: Program Limitations 103
Appendix B: Error Messages 105
Appendix C: Motivation For the Development of the Gated Queue 108
Appendix D: Job Set-Up for Running GERTS GQ 114

Bibliography 115

GERTS GQ Simulator for Systems Simulation

Part I: User's Manual

1. GERTS GQ: INTRODUCTION

1.1 Scope and Development of Simulation of "Q" Networks

To conduct simulation of real world systems on a digital computer, it is necessary to have both deterministic and stochastic characteristics. Network techniques effectively describe operational real-world systems.

GERT is an acronym for Graphical Evaluation and Review Technique. The technique was developed during 1963-1966 by A.A.B. Pritsker and a consortium of seven universities under a NASA contract. It provides a method of simulating and solving a wide range of large scale systems problems (1). This method of modelling allows the analysis of complex systems characterized by a high degree of interaction between system components. The speed of computer simulation methods and increasingly accessible and usable simulation languages provide the basis for quick problem formulation and analysis. Using simulation, it appears feasible to reduce problem processing time to the point that timely solutions can be achieved.

Using GERTS GQ networks, a wide class of queueing systems can be represented in graphical form. The objective is to provide a useful tool for modelling large complex queueing networks by programmers, engineers, and non-programmers--in short--by all who need to conduct simulation studies from a graphical or network point of view, and who are not concerned with detailed programming (2).

The GERTS GQ simulation program exploits the speed and ease of solutions possible with digital simulation of a queueing network. It further provides a new and improved approach to problem formulation, responsive to operational engineering requirements.

1.2 Description of the Simulation Terminology

The GERTS GQ program simulates a network by a "next event simulation," that is, by advancing time from event to event (3). The terminology commonly used in GERTS is:

- Stochastic - random phenomena with predictable long-run regularities.
- GERTS network - graphical network representing each activity in the project as a branch and each event as a node.
- Branch - an activity and/or an information transfer.
- Nodes - decision-making events functionally related by branches.
- Realization - actual occurrence of the event represented by a designated node during performance of the project.
- Release - realization, exactly once, of an activity incident on a node.
- Scheduling an activity - initiating activity along a branch emanating from a node upon realization of the node.
- Network modification - structural alterations made while the simulation is actually in progress.

- Modification numbers - identification of branches whose completion will trigger switching actions restructuring the network.
- Removal - cancellation of all in-progress activities already scheduled into a node immediately upon its release.

1.3 Features of GERTS GQ

A feature unique to the GERTS GQ is its ability to form simple and gated queues at nodes designated as Q-nodes. This feature makes simulation of queueing networks possible. In addition to the simple queueing capability, GERTS GQ has a special queue which can be opened from any other section of the program (4).

The program has the following characteristics:

- The network is capable of self-modification during the course of each simulation experiment.
- Branches incident to a given node can be selectively eliminated from the network once the node has been realized by use of a "removal" operation.
- Several different types of statistical data can be collected at a statistical node subject to the user's choice.
- Node input: The number of releases incident to a node required for realization is readily controlled.
- Node output: Either a DETERMINISTIC or a PROBABILISTIC selection of the output branches emanating from a node is possible.

- An output activity, once scheduled, may require either a fixed time duration or a probabilistically determined time duration for completion.

Apart from "Q-nodes," there are four (4) categories of special purpose nodes available:

- "source" nodes,
- "mark" nodes (these establish reference times),
- "statistical" nodes (collection locations for statistical data) and,
- "sink" nodes.

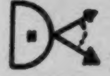
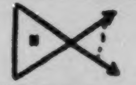
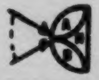
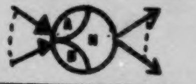
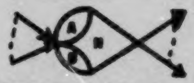
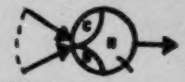
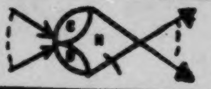
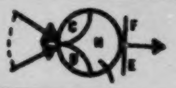
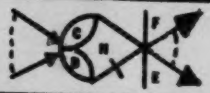
Other nodes appearing throughout the network which do not serve one of the special purposes may by referred to a "regular" nodes.

2. NODES

A node in a stochastic network consists of an input side and an output side. Associated with the input side is the number of activities incident to that node required to realize it. This characteristic is referred to as the number of releases required for the node. It is permissible to specify the number of releases to be less than or greater than the number of activities incident to the node. For example, the number of releases can be two (2), whereas the number of activities incident to the node could be three (3). This would represent the case where if two (2) of the three (3) activities were realized, the node would be realized. Alternatively, the number of releases can be two (2) and the number of activities incident to the node could be one (1). This would represent the case where the activity must occur twice before the node is realized. The GERTS node capabilities are summarized in Table 2.

Table 2

GERTS GQ NODE CAPABILITIES

NODE FUNCTION	DETERMINISTIC SYMBOL	PROBABILISTIC SYMBOL
SOURCE		
SINK*		N/A
EVENT		
QUEUE		
GATED QUEUE		

PARAMETERS:

A = No. of time activities incident to the node must be realized for node to be realized

B = No. of time activities incident to the node must be realized after the first time the node is realized to repeat

C = Initial number in the queue

D = Maximum number allowed in queue

E = Control node number to reopen gate

F = Number of pulses let out at the first time

N = Node number

2.1 Event Nodes

Nodes can be categorized by their function in the network (see section 3. for branches). For network modelling purposes, each node (whether DETERMINISTIC or PROBABILISTIC) can be classified in one or more of the categories summarized below.

2.1.1 SOURCE Nodes:

Nodes which initiate activities at the origin time of the project (time 0). The start event causes all source nodes to be realized and schedules the activities emanating from the source nodes according to the output type of the source node (see Appendix B for the number of source nodes available).

2.1.2 SINK Nodes:

Nodes which may be the terminal node of the network. Normally, SINK nodes only receive flows, but it is also possible to have activities leaving a SINK node if so desired.

2.1.3 STATISTICS Nodes:

STATISTICS nodes are those at which time statistics are collected. There are five (5) basic types of statistics which can be collected. Any node except a SOURCE, QUEUE, or MARK node is a candidate for a STATISTICS node. Statistics are automatically collected on all queue and sink nodes.

At a "statistical node," estimates are computed on the times at which a node is realized. These statistical estimates are compiled over a preassigned number of simulation runs. The mean time to realize the node, its standard deviation, its maximum and its minimum are obtained. A histogram giving the frequency distribution for this time is also assembled.

Table 2.1 of the following page lists the five (5) types of time statistics encompassed.

2.1.4 MARK Node:

This node is used as a time frame reference point for an item being processed. The point in time at which an entity passes a MARK node is recorded as an attribute of that entity. MARK statistics are collected at INTERVAL STATISTICS nodes and constitute the time spent in passing between the MARK node and the INTERVAL STATISTICS node.

TABLE 2.1

TYPE	DESCRIPTION
F	The time of the first realization of a node for each run of the simulation;
A	The total time for all realizations of a node during a given simulation;
B	The time between a realization of a node and the following realization of that node;
I	The time interval required to go between two nodes in the network (from MARK node to STATISTICS node); and
D	The time delay from the first release into the node until its subsequent realization.

(Statistics are automatically collected on all queue and sink nodes.)

2.2 Queue Node

The Q node provides a storage capability for on going items. The characteristics of Q nodes differ sufficiently from those of the other types of nodes occurring in GERTS to justify a separate description.

The concepts of initial and subsequent releases are inappropriate for a Q node. A Q node provides a storage capability for activities which are in progress. The word item is used to represent the idea of a customer, a transaction, etc., which arrives at the node to receive some kind of service activity.

2.2.1 Simple Queue

The simple Q node is characterized by:

- the number of items initially in the queue,
- the maximum number of items allowed in the queue,
- the node to which an item would "balk" (be diverted) if it arrived when the queue was already full.

It is assumed that only one item can be served at a time. If an item is currently being serviced then newly arriving items must "wait in line." Incoming items will automatically be held in the queue until the service activity can be performed.

During operation if a new item arrives at the Q-node when the service activity is unoccupied this service will, of course, be initiated. If the service activity is already occupied, a newly arriving item is put into the queue by increasing the number in the queue counter by one. When this number attains the maximum value permitted for the node, a newly arriving item cannot join the queue but must balk either to another node or from the system entirely (if no balk node has been specified).

The major restriction associated with Q-nodes is that only one service activity can emanate from a Q-node, i.e. there can be only one output branch when the output is deterministic. However, a probabilistic output from a Q-node is allowed where each of the activities emanating from the Q-node can be considered to represent the same server. Other restrictions associated with a Q-node are:

- a service activity cannot be an input to a Q-node (i.e. two Q-nodes cannot be connected by a branch),
- activities which are either incident to a Q-node or represent a service activity cannot have counter types associated with them,
- if the maximum number allowed in the queue is not specified, a large value is assigned. If no queue is really desired, a -1 must be inserted for the maximum number allowed in the queue,
- see Appendix A for other limitations.

Statistics are automatically collected on each Q-node. When the number initially in the queue is non-zero, that number of entities is in the queue and there is one more entity scheduled for the processor. That is, one more pulse than the user specifies leaves the queue.

2.2.2 Gated Queue

The gated Q-node is characterized by:

- the number of items initially in the queue,
- the maximum number of items allowed in the queue,
- the node to which an item would "balk" (be diverted) if it arrived when the queue was already full,
- the event node which opens the gate (signal node),
- the number of items to let out of the queue before closing the gate initially.

The gate may be specified as open or closed initially. Open is specified by a number "n" larger than zero. The gate closes after "n" items leave the queue; each time an item arrives at the signal node, one item may leave the Q and the gate closes again.

If it is desired, the gate may initially be closed by specifying a zero as the number to leave the queue. When an item arrives at the signal node, the gate will open for one item, then close, etc. Each arrival at the signal node lets one more out of the queue.

If a new item arrives at the Q-node when the gate is open, it will proceed. If the gate is closed, it is put into the queue. When the maximum number is reached, newly arriving items must either balk to another node or from the system entirely (if no balk node has been specified).

The same restrictions apply to gated queues as to simple queues. That is, only one activity can emanate from a GQ-node when the output is deterministic. However, a probabilistic GQ node is allowed. Other restrictions are:

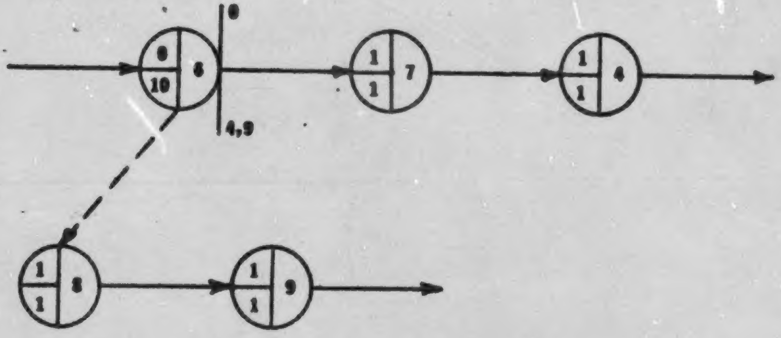
- two (2) queues cannot be connected by a branch,
- activities incident to a GQ-node cannot have counter types associated with them,
- a signal node cannot be used for more than one queue; each GQ must have at least one signal node and it cannot be a node that comes directly after any queue node,
- if the maximum number allowed in the queue is not specified, a large value is assigned; if no queue is really desired, a -1 must be inserted for the maximum allowed in the queue.

- see Appendix A,
- statistics are automatically collected (see Appendix A for maximum number of queues).

Example: In this example, ten entities are collected before the flow of entities begins. In Figure 2.2.2, queue node #6 describes a gate which is initially closed.

- The zero at the top of the gate denotes that initially no entities may leave the queue.
- The entries 4 and 9 at the bottom of the gate define that each time an entity enters either node #4 or node #9 the gate is opened to allow one entity to leave.
- Entities flow into node #6; since the maximum allowed queue length is 10 extra entities balk to node #8.
- From node #8 the entity travels to node #9, a signal node for the gated queue.
- If node #6 were a conventional queue one entity at a time would travel from node #6 to node #7 as long as entities remained in the queue.
- The balk node is also available but it and other nodes are ineligible as signalling nodes.

Figure 2.2.2



2.3 Deterministic and Probabilistic Nodes

If the node is a probabilistic node, only one branch leading from the node will be selected to schedule activities each time the node is realized. The sum of the probabilities of the branches has to equal one (± 0.002). The choice of which branch to use is determined by a random number.

If the node is a deterministic node, every branch emanating from it is scheduled when the node is realized. Thus, if a deterministic node has three (3) branches emanating from it, three (3) entities will be scheduled each time the node is realized.

2.4 Removal

An extremely useful characteristic associated with a GERTS GQ-node is the halting of the on-going activities scheduled into a node once it has been realized. This operation is called "removal" and is indicated by placing the symbol R alongside the release counters A and B on the graphical symbol for the node. The actual removal operations are carried out within the GERTS program itself. A simple example is shown in Figure 2.4A.

Here, two releases cause realization of node 4. The letter R alongside each release identifier signifies that upon completion of two of the activities incident to node 4, it will be realized and all the other activities incident to this node which are currently in process will be automatically cancelled.

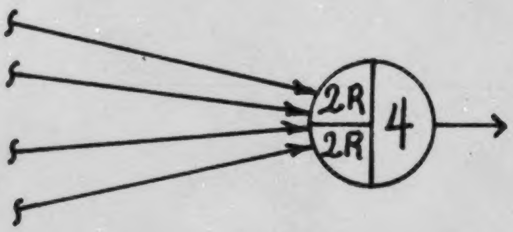


Fig. 2.4A

3. BRANCHES

3.1 Function of Branches and Parameters

A directed branch has associated with it one node from which it emanates and one node at which it terminates.

Characterization by Parameters

- A probability (p) of being included in the network. This is the probability associated with whether or not a particular branch emanating from a node will be taken. This probability is specified when the branch is defined.
- A time required to traverse the branch, i.e. to complete the activity represented by the branch. This traversal time is governed by a distribution whose type and properties are prescribed, when the branch is defined, by specifying a "distribution type number" (t_D) and a set of parameters represented by the "parameter set number" (t_p).

Graphical Representation

The graphical representation of a directed branch, without showing the nodes involved, is given by a directed line identified by parameters (p, t_p, t_D, C, a). The five (5) parameters, which are associated with the branch, are specified by the row vector shown in the sketch. In an actual case numerical values are given each of the parameters.

$$\xrightarrow[(p,\ t_p,\ t_D,\ c,\ a)]{\qquad\qquad\qquad\qquad}$$

Of the following branch specification parameters, the time distribution does require some further clarification. Associated with an activity is a probability that the activity will be realized given that its start node is realized. This probability is specified by giving a numerical value to P.

The branch specification parameters are:

p = probability of realization

t_p = parameter set for time

t_D = distribution type for time

C = counter type (see section 3.4)

a = modification number (see section 3.3)

3.2 Probabilistic Time Distributions

The time required to perform the activity, given that the branch has been taken, is subject to a distribution whose type is specified by t_D. For GERTS GQ, there are nine (9) distributions available in Table 3.2A.

Table 3.2A

t_D	Distribution
1.	Constant
2.	Normal
3.	Uniform
4.	Erlang
5.	Lognormal
6.	Poisson
7.	Beta
8.	Gamma
9.	Beta (fitted to three parameters as in PERT)

When a particular distribution has been selected, the arrangement of the parameters in the corresponding parameter set is automatically specified. These parameters are given numberical values (such as the mean and the standard deviation in the case of the normal distribution, $t_D = 2$) completely defining the properties of the distribution.

Samples are obtained from distributions such that if a sample is less than the minimum value, the sample value is given the minimum value. Similarly, if the sample is greater than the maximum value, the sample value is assigned the maximum value. This is not sampling from a truncated distribution, but sampling from a distribution with a given probability of obtaining the minimum and maximum values.

3.3 Network Modification

A powerful tool when using the GERTS GQ program to analyze complex queueing networks is the ability to modify the network while an activity is in progress. Modification numbers are placed on branches to permit network modifications based on the completion of an activity.

Modification Numbers

Specification of a modification number allows network modifications based on the completion of specified activities within the model. A branch may or may not be numbered. Specification of a modification number does not automatically indicate that the network will be modified. However, only branches with modification numbers can cause the network to be modified. The modification number may be written adjacent to the branch to which it is assigned, and for graphical display is enclosed in a small box.

Network modification involves the replacing of a node by another node on the output side only. Thus, when an entity enters a node, the activities to be started depend on the modifications that have taken place. For example, if node X replaces node Y, then when an entity enters node Y, the activities emanating from node X will be scheduled to start. A node may be replaced and reinserted into the network many times before it is actually realized. It should also be noted that the number of pulses to realize the node is not changed. Therefore, it should usually be the same for the original node as for the replaced node.

Graphically, network modifications are represented by a directed dashed branch. The modification number causing network modification to occur is shown adjacent to that branch enclosed in a square.

3.3.1 An Example of Network Modification

A simple network is shown in Figure 3.3A. Here the branch joining nodes 4 and 5 has been assigned the modification number "1". The dashed branch from node 2 to node 6 with the "1" indicates that when an entity finishes traversing the branch with the "1" (i.e. arrives at node 5), node 2 is replaced with node 6.

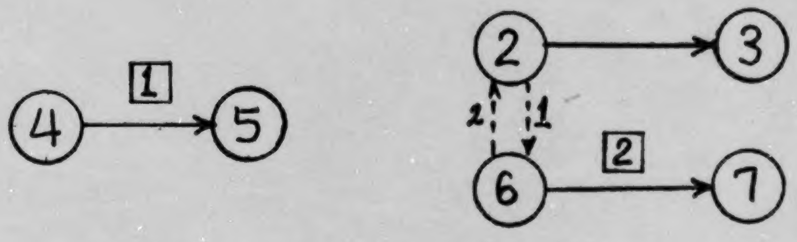


Figure 3.3A

As long as no entity arrives at node 5, this network is equivalent to the network shown in Figure 3.3B.

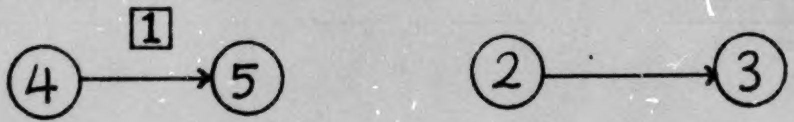


Figure 3.3B

When an entity arrives at node 5, modification occurs and the network becomes equivalent to the one shown in Figure 3.3C.

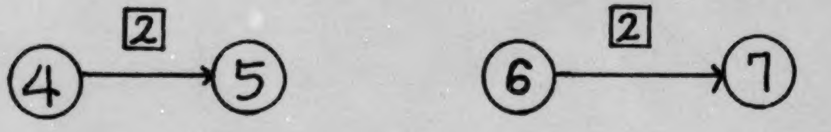


Figure 3.3C

Finally, in this example, the branch from node 6 to node 7 has a modification number 2. When the entity arrives at node 7, the dotted line from node 6 to node 2 indicates that node 6 is replaced by node 2. When this occurs, the actual network in operation is again Figure 3.3B.

Note that modification occurs only when an entity has finished traversing the branches and actually arrives at the end node, not when it leaves the starting node of the branch.

3.4 Count Types

A counter totals every time a branch is used. The "counter type number" identifies the "counter type" associated with a branch. A counter type number specifies the counter to be increased by one each time an entity traverses the branch. Statistics are automatically kept on the counter types for each statistics node. The count for a particular statistics node is finished collecting data for one simulation run after the node is first realized. At the end of all simulation runs, the average and standard deviation of the number of times a counter type branch was traversed prior to the realization of each statistics node is calculated and printed. In addition, minimum and maximum numbers of times branches having the specified counter type were traversed is printed.

To illustrate the use of a counter type we consider the simple network shown in Fig. 3.4A. Here, the branch from node 4 to node 3 has been designated as counter type 1. It is assumed that node 5 is a statistical node, i.e. a node on which statistical data are being collected. The counter type function causes statistics to be automatically collected on the number of times that the branch (4,3) was taken prior to the realization of node 5.

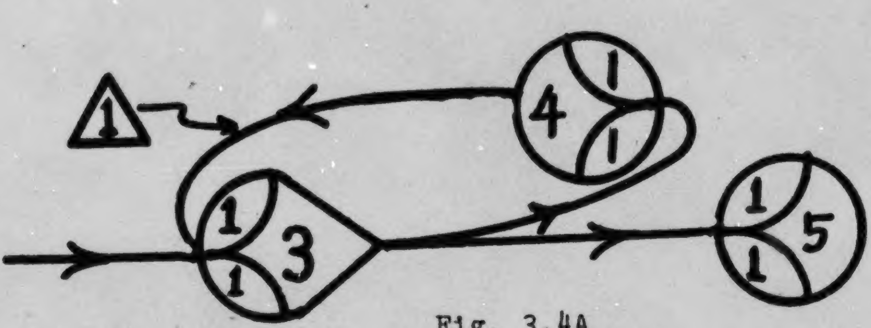


Fig. 3.4A

4. Program input description and format.

Items marked with an asterisk (*) have limitations. See Appendix A.

4.1 Card Type 1: Identification

```
         1            2            3           4          5
123456789012|3456|78|90|1234|5678|9012|3456|78901234|567890|1234
     A1     | A2 |A3|A4| A5 | A6 | A7 | A8 |   A9   |  A10 |
```

Name	Col.	Meaning
A1	1-12	Program name or analyst's name (6A2)
A2	13-16	Program number (if negative, data card 8 is required to indicate the runs to be traced or activity profile to be printed out) (I4)
A3	17-18	Month (right justified) (I2)
A4	19-20	Day (right justified) (I2)
A5	21-24	Year (right justified) (I4)
A6	25-28	Number of times network is to be simulated (right justified) (I4)
*A7	29-32	Number of different time parameters (right justified) (I4)
*A8	33-36	Number of branches in the network plus an estimate of the maximum number of activities which can occur simultaneously (approximately: four times the number of branches) (right justified) (I4)
*A9	37-44	An integer random number seed (right justified) (I8)
A10	45-50	Put 1 in column 50 if A2 is negative and activity profile is requested (I6)

Examples:

```
                  1         2         3         4         5
         12345678901234567890123456789012345678901234567890
Example:              -1008131974  20   2  50    1357     1
Example:              -2008201974  50   3 100    1357
Example:               3008251974 100   7 150    1357
```

Line 1: activity profile

Line 2: trace

Line 3: neither activity profile nor trace

4.2 Card Type 2: Specification

```
         1         2         3
123456789012345678901234567890
B1 B2 B3 B4 B5 B6 B7 B8
```

Name	Col.	Meaning
*B1:	1-3	Largest node number of the network, (smallest number is 2) (right justified) (I3)
*B2:	4-6	Number of source nodes (right justified) (I3)
B3:	7-9	Total number of sink nodes, (right justified) (I3)
B4:	10-12	Number of sink nodes that must be realized before the network is realized (right justified) (I3)
*B5:	13-15	Number of nodes which statistics are to be collected on, including all sink nodes (right justified) (I3)
*B6:	16-18	Number of count types (right justified) (I3)
B7:	19-21	If network modifications exist, put a '1' on column 21, otherwise leave blank (I3)
*B8:	22-24	Number of gated queues, (right justified) (I3) This can be re-interpreted as the number of data cards to be read concerning gated queues.

Examples:

```
         1         2         3
123456789012345678901234567890

 26  1  2  1 17  4  1  3
```

```
 30  1  5  3 20
```

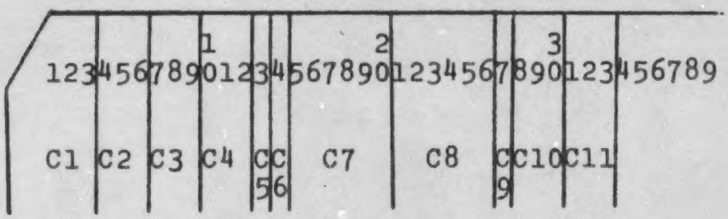
```
 50  1  8  2 35  1  1
```

Line 1: count types, modifications, gated queues

Line 2: no count type, no modifications and no gated queue

Line 3: count type, modifications, no gated queue

4.3 Card Type 3: Description



Name	Col.	Meaning
C1:	1-3	Node number (right justified) (I3)
C2:	4-6	In Col. 6 - special characteristic of the node: 1: Source node 2: Sink node 3: Node on which statistics are collected 4: A mark node 5: Queue node 0 or blank: No special characteristic
*C3:	7-9	Number of releases required to realize the node for the first realization (right justified) (I3) For an event node: Number of releases required to realize the node for the first realization (right justified) (I3) For a Q-node: Initial number in the queue. If it is greater than zero, the service activity is set to be busy and an end of service activity event is defined automatically.
C4:	10-12	Number of releases required to realize the node after the first realization (right justified) For a queue node, -1 indicates maximum number in the queue, 0 indicates no limit, otherwise the maximum number allowable in the queue (right justified) (I3)
C5:	13	Output characteristic of the node. Codes for input are: P for Probabilistic; and D for

		Deterministic. Blank is considered as deterministic (A1)
C6:	14	If events that have been scheduled to end on this node are to be removed (cancelled) when this node is realized, an "R" should be put in this field. If removal is not desired, leave blank (A1)
C7:	15-20	Use only if the node is sink, statistics, or queue node (code 2,3,5 in field 2). The lower limit of the second cell for the histogram to be obtained for this node. The first cell of the histogram will contain the number or times the node was realized in a time less than the value given in this field (F6.2)
C8:	21-26	The width of each cell of the histogram. Each histogram contains 32 cells. The last cell will contain the number of times the node was realized in a time greater than or equal to the lower limit (specified in field 7) + 30 X (cell width specified by Field 8) (F6.2). If negative sign is given a histogram will be plotted. For Queue Node Lower limit of cell 2 and width of each cell for statistics on the average number in the queue.
C9:	27	Statistical quantities to be collected (A1) F. The time of first realizations of the node, A. The time of all realizations of the node, B. The time between realizations of the node, I. The time interval required to go between two nodes, D. The time delay from first activity completion on the node until the node is realized.
C10:	28-30	Use for activity profile purpose. If "no" appears, the activity profile on this node will be suppressed. Otherwise it assumes "yes" (A3)

C11 31-33 For the queue node only, balker node number. The node to which an entity is transferred when an entity enters a Q node and the maximum number allowed is in the queue (the node to which an item balks) (right justified) (I3)

After the last card of this type will be a blank card.

```
         1         2         3
123456789012345678901234567890123456789

  2  1
```

```
 25  2  1  1D
```

Line 1: Source node

Line 2: Sink node, deterministic, with plotter, Activity profile

Line 3: Statistic node, probabilistic, replacable, with plotter, no activity profile

Line 4: Queue node with activity profile

Line 5: Not a specific node---

4.4 Card Type 4: Time Parameters

The parameters associated with the distribution of the time to perform each activity. One card is required for each activity with a different time characterization. The number of cards is specified by data card 1, Field 7.

```
         1         2         3         4
1234567890123456789012345678901234567890123 45

    D1        D2        D3        D4
```

The cards must be arranged by ascending parameter number and the parameters must be numbered consecutively or blank cards appropriately placed. Nine distribution types are available which are:

1. Constant
2. Normal
3. Uniform
4. Erlang
+5. Lognormal
6. Poisson
+7. Beta
+8. Gamma
+9. Beta fitted to three parameters as in PERT

+ The fields required are dependent on the distribution type of the activity. When using these distributions, use the time parameter for only one branch. The time parameters are changed in the program.

For Distribution Type 1 (constant):

D1 Col 1-10 The constant time (F10.4).
D2, D3, D4 are not used.

For Distribution Type 2 (normal), 5 (lognormal), 7 (beta), and 8 (gamma):

Name	Col.	Meaning (contents)
D1	1-10	The mean value (F10.4)
D2	11-20	The minimum value (F10.4)
D3	21-30	The maximum value (F10.4)
D4	31-40	The standard deviation (F10.4)

For Distribution Type 3 (uniform):

Name	Col.	Meaning (contents)
D1	1-10	Not used (F10.4)
D2	11-20	The minimum value (F10.4)
D3	21-30	The maximum value (F10.4)
D4	31-40	Not used (F10.4)

For Distribution Type 4 (erlang):

Name	Col.	Meaning (contents)
D1	1-10	The mean time for the Erlang variable divided by the value given to Field 4 (F10.4)
D2	11-20	The minimum value (F10.4)
D3	21-30	The maximum value (F10.4)
D4	31-40	The number of exponential deviates to be included in the sample obtained from the Erlang distribution (F10.4)

If field 4 is set equal to 1, an exponential deviate will be obtained from distribution type 4.

For Distribution Type 6 (poisson):

Name	Col.	Meaning (contents)
D1	1-10	The mean minus the minimum value (F10.4)
D2	11-20	The minimum value (F10.4)
D3	21-30	The maximum value (F10.4)
D4	31-40	Not used

Care is required when using the Poisson since it is not usually used to represent an interval of time. The interpretation of the mean should be the mean number of time units per time period.

For Distribution Type 9 (Beta fitted to 3 values as in PERT):

Name	Col.	Meaning (contents)
D1	1-10	The most likely vlaue, m (F10.4)
D2	11-20	The optimistic value, a (F10.4)
D3	21-30	The pessimistic value, b (F10.4)
D4	31-40	Not used

Example:

```
         1         2         3         4
1234567890123456789012345678901234567890

    10
```

```
         1         2         3         4
1234567890123456789012345678901234567890

     5         0        10         2
               1        20
     250       0        10         2
    10         5        30
     520       1        15
```

Line 1: type 1

Line 2: type 2,5,7,8

Line 3: type 3

Line 4: type 4

Line 5: type 6

Line 6: type 9

4.5 Card Type 5: Gated Queue

At least one card is required for each gated queue, the number has been specified in field 8 card type 2 (B8)

```
         1
1234567890123456789

E1 E2 E3
```

Name	C	Col.	Meaning (contents)
E1		1-3	Gated Queue node number (right justified) (I3)
E2		4-6	Control node number to reopen gate, everytime this node is entered, the gated queue will open and release one entity (right justified) (I3)
E3		7-9	Number of entities initially released before gate is closed (right justified) (I3)

Two queues may not be opened by the same signal node.

However, one queue may be opened by any number of signal nodes.

```
          1
1234567890

  4  7  2
  3 65  1
  4  8  2
  4  6  2
  5 14  5
 10 12  0
```

Lines 1,3,4: Node number 4 is a gated queue which initially lets two out of the queue before gating. It opens for one more each time node 7, node 8, or node 6 has a pulse enter it. A maximum of fifty such cards can be read.

Line 2:
Gated queue node 3
Control node 65
First release 1

Line 5:
Gated queue node 5
Control node 14
First release 5

Line 6:
Gated queue node 10
Control node 12
Gate is initially closed

4.6 Card Type 6: Branching Activity

One data card for each associated with the network.

```
        |  1|   |   |  2|   |   |  3
12345678|901|234|567|890|123|456|7890
   F1   |F2 |F3 |F4 |F5 |F6 |F7 |
```

Name	Col.	Meaning (contents)
F1	1-8	Probability of realization (F8.3)
F2	9-11	Start node (right justified) (I3)

F3	12-14	End node (right justified) (I3)
F4	15-17	Parameter number (right justified) (I3)
F5	18-20	The distribution type (right justified) (I3)
F6	21-23	Count type (right justified) (I3)
F7	24-26	Activity number (right justified) (I3)

The last data card of this type <u>must</u> have a zero (or blank) in field 2.

Example:

```
         1         2         3
123456789012345678901234567890
    1.    2  3  1  1
    1.    4  5  2  3  3
    1.    7  8  2  3  2  1
     .70  9 10  2  1
```

Line 1: no type count, no activity number

Line 2: count type

Line 3: activity number

Line 4: probabilistic path---

4.7 Card Type 7: Network Modification

Only required if field B7 of data card 2 is greater than zero.

```
         1         2         3         4         5
123456789012345678901234567890123456789012345678901
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10G11G12G13G14G15G16G17
```

Name	Col.	Meaning (contents)
G1	1-3	An activity number (right justified) (I3)
G2	4-6	The number of the node to be replaced if the modification given in field 1 is realized (right justified) (I3)
G3	7-9	The number of the node to be inserted into the network in place of the node specified in field 2 when the modification in field 1 is realized (right justified) (I3)
G4-G 21	++	Fields 2 and 3 are repeated if the activity given in field 1 affects multiple nodes. A zero in an even-numbered field indicates the end of the data on the card.

The last card of this type must have a zero in field 1.
++ Three columns each, (right justified), if needed, leave blank otherwise.

Example:

```
         1         2         3         4
1234567890123456789012345678901234567890

  1  9 10 20 21
  2 10  9 21 20
  3  7  8
  4  8  7
```

Line 1: when modification 1 occurs, replace node 9 with node 10 and replace node 20 with 21

Line 2: when modification 2 occurs, replace node 10 with node 9, and node 21 with node 20

Line 3: when modification 3 occurs, replace node 7 with node 8

Line 4: when modification 4 occurs, replace node 8 with node 7

4.8 Card Type 8: Trace or Activity Profile

This card is only used if the project number is negative, (field A2, Data Card 1).

```
         1         2
12345678901234567890

 H1 H2
```

Name	Col.	Meaning
H1:	1-3	The run number for which tracing or activity profile should begin, (right justified) (I3)
H2:	4-6	The run number for which tracing or activity profile should terminate, (right justified) (I3)

Examples:

```
         1         2
12345678901234567890

  1  1
  1  3
  5  7
```

Line 1: Trace or activity profile is requested for the first run

Line 2: Trace or activity profile for runs one, two and three

Line 3: Trace or activity profile for runs five, six and seven

4.9 Multi-data Sets

Multiple networks can be analyzed by stacking the data cards as described above, one after another. No blank cards should separate the data cards for each network. A blank card is required to indicate the end of all networks to be simulated.

5. Program Output

Program output takes place sequentially in three cases which are separately discussed:

- a data input check or "echo check" (sec. 5.1),
- a "trace" or an "activity profile"(both user specified options showing the propagation of events through the network during running (sec. 5.2, 5.3), and
- statistical summaries compiled for end-of-simulation for (sec. 5.4, 5.5).

5.1 Data input check

This stage of the program output, often called an "echo check", presents the network parameters actually used during the run, permitting their verification. The examples below illustrate the kinds of information displayed for each type of data card.

5.1.1 Data card types 1 and 2

```
                    GERT SIMULATION PROJECT-660 BY PHASE-3
                                 DATE  8/  7/ 1974

                              **NETWORK DESCRIPTION**

                               NODE CHARACTERISTICS

HIGHEST NODE NUMBER IS   26
NUMBER OF SOURCE NODES IS    1
NUMBER OF SINK NODES IS    2
NUMBER OF NODES TO REALIZE THE NETWORK IS     1
STATISTICS COLLECTED ON    5 NODES
NUMBER OF PARAMETER SETS IS   10
INITIAL RANDOM NUMBER IS    1237          0
```

5.1.2 Data card type 3

```
NODE  NUMBER  NUMBER OF RELEASES OUTPUT REMOVAL DESIRED  STATISTICS BASED
     RELEASES     FOR REPEAT      TYPE   AT REALIZATION  ON REALIZATIONS

   2       0            9999         D
   3      -1              -1         D
   4       1               1         D                          A
   5       1               1         D                          B
   6       1               1         P
   7       1               1         P                          A
   8       1               1         D                          A
   9       1               1         D                          B
  10       1               1         D                          A
  11       1               1         D                          B
  12       1               1         P
  14       1               1         D            R

                             QUEUE NODES

          NODE    INITIAL =   MAXIMUM =   OUTPUT   NODE FOR    PRIORITY
                  IN QUEUE    ALLOWED      TYPE    BALKERS      SCHEME

           16        0          9999        D         0          FIFO
           21        0          9999        D         0          FIFO

                    SOURCE NODE NUMBERS
                          2

                    SINK NODE NUMBERS
                          7

                    STATISTICS COLLECTED ALSO ON NODES
                         24   19   12
```

5.1.3 Data card type 4

```
                          **ACTIVITY PARAMETERS**

PARAMETER                                      PARAMETERS
 NUMBER               1              2              3              4

    1                 0              0              0              0
    2             .2500          .1600          .3000          .0200
    3            1.0000          .8000         1.1200          .1000
    4           13.0000        10.0000        16.0000         1.0000
```

5.1.4 Data card type 5

```
                          **GATED QUEUES**

   QUEUE            SPECIAL             NUMBER
   NODE             NODE                LEAVING
      16                  14                  3
```

5.1.5 Data card type 6

```
                        **ACTIVITY DESCRIPTION**

 START  END  PARAMETER       DISTRIBUTION     COUNT  ACTIVITY  PROBABILITY
  NODE  NODE   NUMBER            TYPE          TYPE   NUMBER

   2     3       4                2             0        0        1.0000
   2     3       3                2             0        0        1.0000
   2     3       3                2             0        0        1.0000
   3     4       4                2             0        0        1.0000
   4     5       5                2             0        0        1.0000
   5     6       6                1            -1        0        1.0000
   5     9       6                1             0        1        1.0000
   6     7       7                1            -6        0        1.0000
```

5.1.6 Data card type 7

```
                            **NETWORK MODIFICATIONS**
ACTIVITY NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE NODE

   1        7    8
   2       10   11
```

5.2 Trace

The user may specify that a trace (or an activity profile) form part of the program output. The "trace", a useful tool for checking the validity of a system, documents the sequence of events as they occur throughout the network. The trace, obtained while a simulation is being run, provides a step-by-step description of the user's program. The trace does have two deficiencies. The First--after a system has been modified, the node numbers printed out are still the original node numbers because the input parts of nodes are not changed, only their output sides. Second--there is no direct way to determine how many entities are in a node; manual tally is required whenever an entity enters a node.

A typical trace from the old GERTS III Q program is shown in Fig. 5.2A. In the new GERTS GQ program "WAS REALIZED" is replaced with "OCCURRED". The older phrase was misleading because the node was not actually realized; what was meant was that an entity had entered the end node for that branch. Consider the figure: time 50.87, node 10 has an entity entering it. The numbers to the right of the word ATTRIBUTES signify that time parameter 1 is used with distribution type 1 (constant). The zero in the third column denotes that Node 10 is not a special node. A -271 in this column denotes a delay node; a -1, a queue node (i.e. nodes 16 and 21). A signal node controlling a gated queue would have the negative of the gated queue node's number in this column. The incoming branch for this node is not special either. If there were a negative queue node number in the third column, it would be a processor for a queue, or the end node would be a signal node (#13) and if it were a positive number, the branch would have had a counter type on it (i.e. nodes 6 and 9). The last column has a zero in it indicating that no modification number was associated with this node. This trace was taken on run number 1. From the print-out, the user can determine which event took place first. At time 70.49, three activities occurred simultaneously, but the sequence shows that node 6 was entered first, when 7 and finally 10. This capability, being able to determine which activity happened first, helps to make the trace a powerful tool for program analysis and diagnosis.

Fig. 5.2A

```
AT TIME      50.87 ACTIVITY ON NODE    10 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      51.66 ACTIVITY ON NODE     3 WITH ATTRIBUTES     2      3      3       0 WAS REALIZED ON RUN   1
AT TIME      51.66 ACTIVITY ON NODE     5 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      54.30 ACTIVITY ON NODE    14 WITH ATTRIBUTES     4      1      0       0 WAS REALIZED ON RUN   1
AT TIME      54.30 ACTIVITY ON NODE    16 WITH ATTRIBUTES     1      1     -1       0 WAS REALIZED ON RUN   1
AT TIME      55.49 ACTIVITY ON NODE     9 WITH ATTRIBUTES     4      3      3       0 WAS REALIZED ON RUN   1
AT TIME      55.49 ACTIVITY ON NODE    10 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      55.54 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      55.54 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      55.54 ACTIVITY ON NODE    21 WITH ATTRIBUTES     1      1     -1       1 WAS REALIZED ON RUN   1
AT TIME      58.01 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      58.01 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      58.01 ACTIVITY ON NODE     8 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      58.37 ACTIVITY ON NODE     4 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      58.37 ACTIVITY ON NODE     9 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      60.67 ACTIVITY ON NODE     3 WITH ATTRIBUTES     2      3      3       0 WAS REALIZED ON RUN   1
AT TIME      60.67 ACTIVITY ON NODE     5 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      61.82 ACTIVITY ON NODE     3 WITH ATTRIBUTES     2      3      3       0 WAS REALIZED ON RUN   1
AT TIME      61.82 ACTIVITY ON NODE     5 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      63.46 ACTIVITY ON NODE    13 WITH ATTRIBUTES     3      3    -16       0 WAS REALIZED ON RUN   1
AT TIME      63.46 ACTIVITY ON NODE    16 WITH ATTRIBUTES     1      1     -1       0 WAS REALIZED ON RUN   1
AT TIME      67.30 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      67.30 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      67.30 ACTIVITY ON NODE    10 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      68.21 ACTIVITY ON NODE     4 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      68.21 ACTIVITY ON NODE     5 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      69.81 ACTIVITY ON NODE    13 WITH ATTRIBUTES     3      3    -16       0 WAS REALIZED ON RUN   1
AT TIME      69.81 ACTIVITY ON NODE    16 WITH ATTRIBUTES     1      1     -1       0 WAS REALIZED ON RUN   1
AT TIME      69.93 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      69.93 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      69.93 ACTIVITY ON NODE    21 WITH ATTRIBUTES     1      1     -1       1 WAS REALIZED ON RUN   1
AT TIME      70.49 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      70.49 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      70.49 ACTIVITY ON NODE    10 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
AT TIME      73.54 ACTIVITY ON NODE     6 WITH ATTRIBUTES     3      3      3       0 WAS REALIZED ON RUN   1
AT TIME      73.54 ACTIVITY ON NODE     7 WITH ATTRIBUTES     1      1      0       0 WAS REALIZED ON RUN   1
```

5.3 Activity profile

Instead of a trace the user of GERTS GQ may specify an "activity profile" be printed-out. The activity profile provides a graphical trace of the simulation run while it is in progress(5). Node numbers are listed across the page, using two spaces per node, times are listed down its vertical edge. Two lines are printed for each time unit. The first line shows the content of each node. Symbols used to represent how many entities or in a node are:

.-no entities in node
T-one entity is in node
2-99-the number printed is the number in the node.
A-100-199 entities
B-200-299 entities
C-300-399 entities
D-400-499 entities
E-500-599 entities
F-600-699 entities
G-700-799 entities
H-800-899 entities
I-900-999 entities

In the second line, symbols printed underneath each node number denote the entry or exit of an entity for that node. Two symbols are used:

↑-an entity enters the node
↓-an entity exits the node

Downward arrows are always printed in the right space of the two spaces reserved for each node; upward arrows in the left space.

The activity profile is used when detailed information about the internal state of a node (i.e. the number of releases at a given time)is desired. This number is important when identifying bottlenecks as well as when verifying the user's model. The output symbols used for an activity may vary from machine to machine. Consult your computer center for the printer's character set.

To illustrate the utility of the activity profile consider the Figure 5.3.A . The figure shows where entity stoppages occur. E.G. node 22 at time 66.9: there are 98 entities in this node; at the time 67.3, another entity enters, and finally, at time 67.9, the 100th entity arrives. Again, consider nodes 33-35; at time 67.1, there is a great deal of activity here. An entity enters node 33 and leaves it again; a pulse enters node 34, which is realized; the pulse leaves again; finally, node 35 acquires an entity. As can be seen, there is one draw back, the user cannot tell from where an entity has originated or what its destination might be.

Figure 5.3.A

```
RUN NO.    1TIME   6 7 816171819212223242526273334353639414344464849515253
         66.900  6667  . 6 . 4 . .98 . . . . . . 3 .13 . .15 . . . .1415 .
                          ↑↓↑                   ↑
         67.000  6667  . 6 . 4 . .98 . . . . . . 4 .13 . .15 . . . .1415 .
                                                ↑
         67.000  6768  . 6 . . T .98 . . . . . . 4 .13 . .15 . . . ,1415 .
                  ↑ ↑     ↑↓↑↓↑
         67.100  6768  . 5 . T . .98 . . . . . . . T13 . .15 . . . .1415 .
                         ↓↑↓↑  ↓              ↑↓↑↓↑
         67.186  6768  . 5 . T . .98 . . . T . . . T13 . .15 . . . .1415 .
                                          ↑
         67.200  6768  . 5 . T . .98 . . . . . . T T13 . .15 . . . .1415 .
                                           ↓    ↑
         67.300  6768  . 5 . T . .99 . . . . . . T T13 . .15 . . . .1415 .
                                ↑↓↑
         67.400  6768  . 5 . T . .99 . . T . . . T T13 . .15 . . . .1415 .
                                        ↑
         67.600  6768  . 5 . T . .99 . . . . . . T T13 . .15 . . . .1415 .
                                         ↓
         67.700  6768  . 6 . 2 . .99 . . . . . . T T13 . .15 . . . .1415 .
                        ↑ ↑↓↑
         67.743  6768  . 6 . 2 . .99 . . . . . . T 213 . .15 . . . .1415 .
                                              ↑↓  ↑
         67.800  6768  . 6 . 3 . .99 . . . . . . 2 213 . .15 . . . .1415 .
                          ↑↓↑                   ↑
         67.900  6768  . 6 . 3 . .A0 . . . . . . 2 213 . .15 . . . .1415 .
                                ↑↓↑
         67.900  6768  . 6 . 4 . .A0 . . . . . . 3 213 . .15 . . . .1415 .
                          ↑↓↑                   ↑
         68.000  6769  . 6 . 4 . .A0 . . T . . . 3 213 . .15 . . . .1415 .
                                        ↑
```

5.4 Statistics

The third stage of program output consists of summaries of statistical data collected on counter types and at all monitored nodes (user-specified). These summaries can furnish valuable performance measures for the system. Therefore, the type and location of statistics nodes specified should be considered carefully.

5.4.1 Statistics nodes

Statistics nodes are essential to the understanding and utilization of a system. These nodes collect information which will enable the user to measure system performance and appraise what improvements can be made. Statistical measures given for a statistics node are probability a node will be realized during a run, and the mean, the standard deviation, the number of observations, and the minimum and maximum of the times for the type of statistics collected.

In the example below, node 16 was realized every time the system was run. The mean time for realization was 53.17, for the "first time" statistics, with a standard deviation of .8694 and a range of 50.51 to 55.23 for 51 observations.

```
                       GERT SIMULATION PROJECT-660 BY PHASE- 3
                                   DATE  8/  7/ 1974


                       **FINAL RESULTS FOR   51 SIMULATIONS**
NODE   PROB./COUNT      MEAN    STD.DEV.      = OF    MIN.       MAX.  NODE TYPE
                                              OBS.


16   1.0000            53.1678    .8694        51.   50.5146   55.2272    A
17             0           NO VALUES RECORDED

13   1.0000            48.1572    .8753        51.   45.5287   50.2380    F

 5   1.0000            16.0287    .8717        51.   13.4365   18.4705    F
```

5.4.2 Counter types

A counter type number specifies the counter which is to be increased every time a branch with a counter type is used. Although the number of branches which can be associated with each counter is unlimited, only one counter can be associated with each branch. At the end of all the runs, the average, standard deviation, number of observations, minimum and maximum for the number of times a counter type was activated will be tabulated for the interval prior to the realization of each of the statistic nodes specified. The number of counts is always referenced by the number of counts prior to the realization of each node which statistics are collected on.

A good example is the counters on node 5. Counters 1 and 3 were activated prior to the realization of node 5. For counter 1, on the average, the counter was activated 5.45 times before node 5 was realized. As can be seen, the standard deviation is 4.28 with a range of 1 to 20. The type of statistics collected has no bearing on the counters.

```
6   1.0000              7.4540     1.4354      3385.    5.0016     9.9981    I
6              1             0          0      3385.         0          0
6              2             0          0      3385.         0          0
6              3       27.1383    18.0831      3385.    1.0000    93.0000

5   1.0000             52.6801    37.2331      3653.         0   164.2996    A
5              1        5.4517     4.2762      3653.    1.0000    20.0000
5              2             0          0      3653.         0          0
5              3       25.0980    18.7567      3653.         0    94.0000
```

5.4.3 Queue node statistics

For each queue, two types of statistics are kept; the average number in the queue, and the average number of balkers per unit time. For the simple queue, a third type is given, the average busy time of processor. The mean, standard deviation, number of observations, minimum and maximum is given for all types of statistics. The mean for each run is calculated by integrating over time and then dividing the resulting quantity by the total time of each run. Then the mean for all the runs is calculated by taking the average of the means; the standard deviation is the standard deviation of the means. The number of observations is the number of runs.

In Figure 5.4.3A, node 21 is a simple queue, while node 16 has a gate. The average number of entities in node 21 was 4.97, indicating that there was a waiting line. The processor was busy almost all the time. At times, there was an overflow of entities from the queue, about one every 12 time units. These figures can be useful in scheduling resources.

Figure 5.4.3A

```
                       **QUEUE NODES**

NODE    MEAN STD.DEV. = OF    MIN.     MAX.
                      OBS.


  21  4.9678   .1506  100.   4.6556  5.2957  AVERAGE NUMBER IN THE QUEUE
  21   .9912   .0074  100.    .9623  1.0000  AVERAGE BUSY TIME OF PROCESSOR
  21   .0827   .0014  100.    .0797   .0875  AVERAGE BALKERS PER UNIT TIME


  16 22.5000       0    1.  22.5000 22.5000  AVERAGE NUMBER IN THE QUEUE
  16       0       0    1.        0       0  AVERAGE BALKERS PER UNIT TIME
```

5.5 Histograms

Histograms show frequency distributions of the realization times for event nodes and for the lengths of queues at queue nodes. The first and last cells are used for overflows. The lower limit for the second cell, the cell width, are provided with all histogram outputs. If a graph is printed, the number of realizations each asterisk represents is printed. The number in each cell is printed, then the cell number and the actual graph. Each bar shows a percentage of the maximum number in a cell. When no values are recorded, there will be no heading. If no graph is shown, the frequencies in each cell are printed. The cells go from left to right in increasing order.

Figure 5.5A shows the frequency of each cell. The cell width and lower limits are 1, but they need not be. The frequency of realization is from time unit 1-2, 5 times, from 2-3, 10, from 3-4, 20 and so on. From these frequencies, we can calculate what kind of distribution the times of realization follows, depending on statistic type.

Figure 5.5B gives an example of the histogram feature of GERTS GQ. For node 6, the lower limit was .6 time units and the cell width was .05. Each asterisk represents 5.89 realizations. The histogram automatically scales the graph so that a maximum of 100 asterisks is shown for any one bar.

Figure 5.5B

```
                    HISTOGRAM FOR NODE NO.    5, THE LOWER LIMIT OF THE SECOND CELL =      .6800

                    THE CELL WIDTH =      .0500    EACH ASTERISK REPRESENTS     5.890 OF A REALIZATION  .

       CELL         1         2         3         4         5         6         7         8         9         0
FREQ.  NO. 1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
    3    1
  152    2 *************************
  294    3 *************************************************
  498    4 ************************************************************************************
  589    5 ****************************************************************************************************
  493    6 ***********************************************************************************
  412    7 *********************************************************************
  235    8 ***************************************
  123    9 ********************
   96   10 ****************
    0   11
    0   12
    0   13
    0   14
    0   15
    0   16
    0   17
    0   18
    0   19
    0   20
    0   21
    5   22
   10   23 *
   20   24 ***
   21   25 ***
   17   26 **
   13   27 **
    7   28 *
    5   29
    2   30
    0   31
    0   32
```

Figure 5.5A

```
HISTOGRAM FOR NODE NO.    4, THE LOWER LIMIT OF THE SECOND CELL =     1.0000

THE CELL WIDTH =    1.0000

          ***FREQUENCIES***

       0      5     10     20     36     36     41     25     20     30     37
      40     35     40     26     32     20     22     31     13     10     10
       9     11      4      7      0      4      0      2      1      0
```

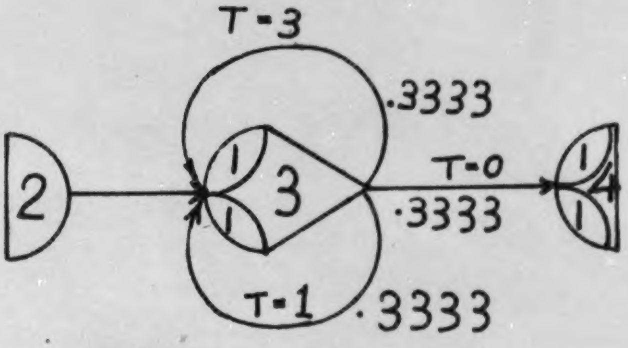
6. Illustrative Examples

These examples serve as a guide to show some of the possible types of networks, as well as a vehicle for explanation of the features of the GERTS GQ program.

6.1 Thief of Bagdad

A thief was trapped in a cave, where he found three tunnels leading out of the cave. The first led directly to freedom, the second went back to the same cave after a day's journey, and the third, again, to the same cave, only this time taking three days. The thief has no memory of the tunnels he took before. Therefore no matter how many times he returns to the cave, he again has an equal probability of choosing each tunnel. How many days will be his average time of escape to freedom? (By statistical methods, the average is four days (6).) Figure 6.1A shows a nodal diagram of the system.

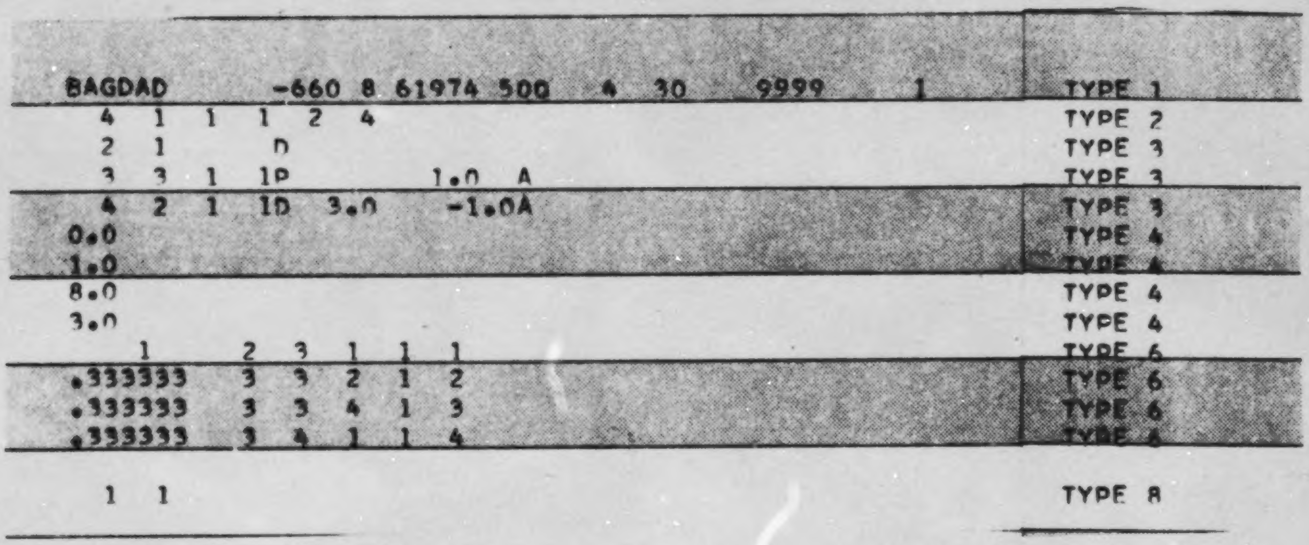
Figure 6.1A



Node 2 is the source node, which emits the thief to start the run. The branch 2-3 is traversed in zero time. At node three, the thief has an equal probability of traversing the three outgoing branches. The only branch which will terminate the run goes to the sink node (branch 3-4, t=0).

Figure 6.1B shows the coding for the data cards. Figure 6.1C shows a computer run of Example 1. The network was simulated 500 times. The average time for node 4 to be realized was 3.93 time units (days), which is comparable to the theoretical value of 4.00.

Figure 6.1B



```
BAGDAD        -660 8 61974 500    4  30     9999       1             TYPE 1
  4  1  1  1  2  4                                                   TYPE 2
  2  1     D                                                         TYPE 3
  3  3  1  1P         1.0  A                                         TYPE 3
  4  2  1  1D  3.0    -1.0A                                          TYPE 3
0.0                                                                  TYPE 4
1.0                                                                  TYPE 4
8.0                                                                  TYPE 4
3.0                                                                  TYPE 4
     1      2  3  1  1  1                                            TYPE 6
.333333     3  3  2  1  2                                            TYPE 6
.333333     3  3  4  1  3                                            TYPE 6
.333333     3  4  1  1  4                                            TYPE 6

  1  1                                                               TYPE 8
```

While count types were not needed in this problem, statistics were gathered on each branch to show what kind of data is gathered. Each branch in the network was traversed approximately one time before node 4 was realized.

Figure 6.1C

```
                                   **NETWORK DESCRIPTION**

                                    NODE CHARACTERISTICS

              HIGHEST NODE NUMBER IS     4
              NUMBER OF SOURCE NODES IS     1
              NUMBER OF SINK NODES IS     1
              NUMBER OF NODES TO REALIZE THE NETWORK IS     1
              STATISTICS COLLECTED ON    2 NODES
              NUMBER OF PARAMETER SETS IS    4
              INITIAL RANDOM NUMBER IS     9999    1.0000

NODE    NUMBER      NUMBER OF RELEASES   OUTPUT     REMOVAL DESIRED  STATISTICS BASED
       RELEASES        FOR REPEAT        TYPE       AT REALIZATION   ON REALIZATIONS

  2       0               9999            D
  3       1                  1            P                                 A
  4       1                  1            D                                 A

              SOURCE NODE NUMBERS
                   2

              SINK NODE NUMBERS
                   4

              STATISTICS COLLECTED ALSO ON NODES
                   3


                           **ACTIVITY PARAMETERS**

 PARAMETER                                   PARAMETERS
  NUMBER               1              2              3              4

     1                   0              0              0              0
     2              1.0000              0              0              0
     3              8.0000              0              0              0
     4              3.0000              0              0              0

                             **ACTIVITY DESCRIPTION**

 START      END      PARAMETER      DISTRIBUTION       COUNT      ACTIVITY       PROBABILITY
 NODE       NODE      NUMBER            TYPE            TYPE       NUMBER

   2         3          1                1               1           0              1.0000
   3         3          2                1               2           0               .3333
   3         3          4                1               3           0               .3333
   3         4          1                1               4           0               .3333

                       GERT SIMULATION PROJECT-660 BY BAGDAD
                                DATE  8/  6/ 1974


                     **FINAL RESULTS FOR  500 SIMULATIONS**

             NODE   PROB./COUNT      MEAN     STD.DEV.     # OF    MIN.        MAX.  NODE TYPE
                                                           OBS.

               4   1.0000           3.9340     5.1876      500.        0   40.0000     A
               4               1    1.0000          0      500.   1.0000    1.0000
               4               2     .9280     1.3938      500.        0   13.0000
               4               3    1.0020     1.4304      500.        0    9.0000
               4               4    1.0000          0      500.   1.0000    1.0000

               3   1.0000           4.0778     5.3180     1465.        0   40.0000     A
               3               1    1.0000          0     1465.   1.0000    1.0000
               3               2     .9857     1.5556     1465.        0   13.0000
               3               3    1.0307     1.4304     1465.        0    9.0000
               3               4         0          0     1465.        0         0
```

```
                  HISTOGRAM FOR NODE NO.    4, THE LOWER LIMIT OF THE SECOND CELL =     3.0000

                  THE CELL WIDTH =     1.0000     EACH ASTERISK REPRESENTS     2.440 OF A REALIZATION

       CELL          1         2         3         4         5         6         7         8         9         0
FREQ.   NO. 1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
  244     1 ****************************************************************************************************
   59     2 ************************
   39     3 ***************
   21     4 ********
   31     5 ************
   17     6 ******
   13     7 *****
   13     8 *****
   14     9 *****
   10    10 ****
    7    11 **
    6    12 **
    3    13 *
    4    14 *
    3    15 *
    3    16 *
    1    17
    3    18 *
    0    19
    1    20
    3    21 *
    1    22
    0    23
    0    24
    0    25
    1    26
    1    27
    1    28
    0    29
    0    30
    0    31
    1    32


                  HISTOGRAM FOR NODE NO.    3, THE LOWER LIMIT OF THE SECOND CELL =          0

                  THE CELL WIDTH =     1.0000

                            ***FREQUENCIES***

                         0    500    162     47    194    118     57     83     60     37     39
                        37     27     23     16      8     12     13      6      5      8      3
                         4      5      3      3      1      2      3      2      2      5
```

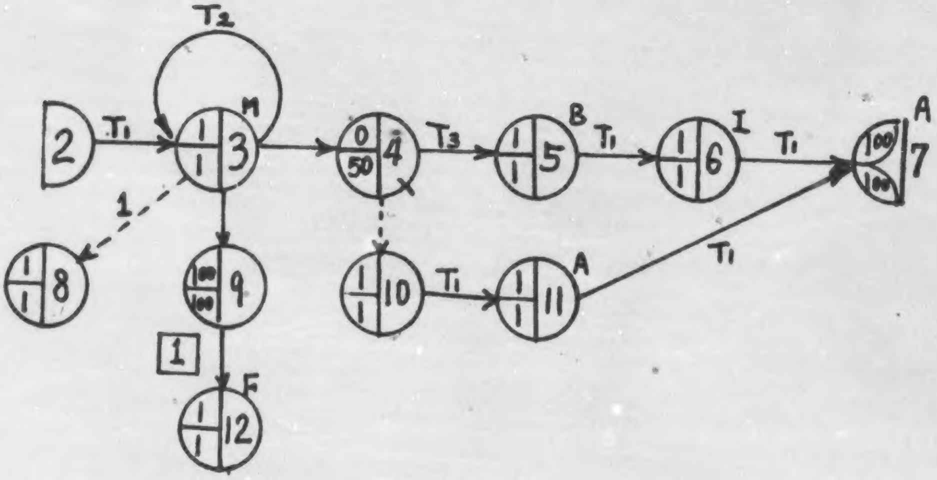
Figure 6.1C continued

6.2 Ticket Box Office Problem

People arrive at a uniform rate between 0 and 2 minutes. They wait in a line, with a maximum length of fifty people waiting. Time spent with the cashier is between one and three minutes.

1. Measure the average length of the line,
2. Measure the time spent in the system, min., max., average,
3. Simulate for one hundred people.

Figure 6.2A

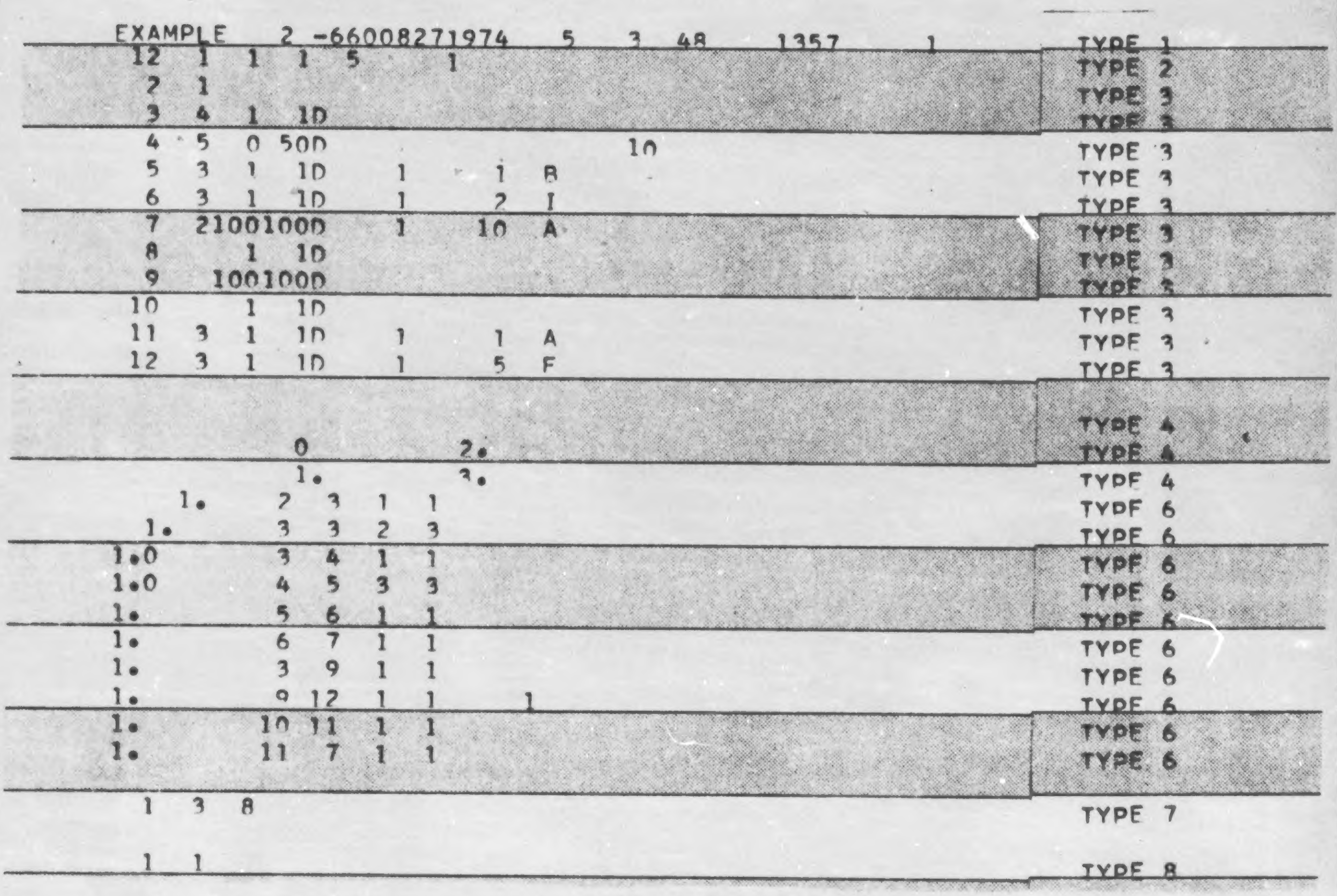


The combination of nodes 3,8,9,12 generates a "person" between 0 and 2 time units for a total of 100 people. After the "people" are generated, they form a queue at the teller's window, queue node 4. The time for the processor (branch 4-5), is the time each person takes in obtaining a ticket from the teller. A maximum of fifty people can wait in the line. When fifty people are waiting, each incoming person will be discouraged and decide that the line is too long (or in other words, he "balks"), so the person will leave the system by way of node 10. Statistics are gathered on:

- how long it took to generate 100 people (node 12,F),
- the time between each person getting past the teller (node 5,B),
- how many people left the waiting line (node 11,A),
- how long it took for each person to get through the system (node 6,I),
- how long the entire system took to run (node 7,A).

Figure 6.2B shows the data cards used for the model.

Figure 6.2B

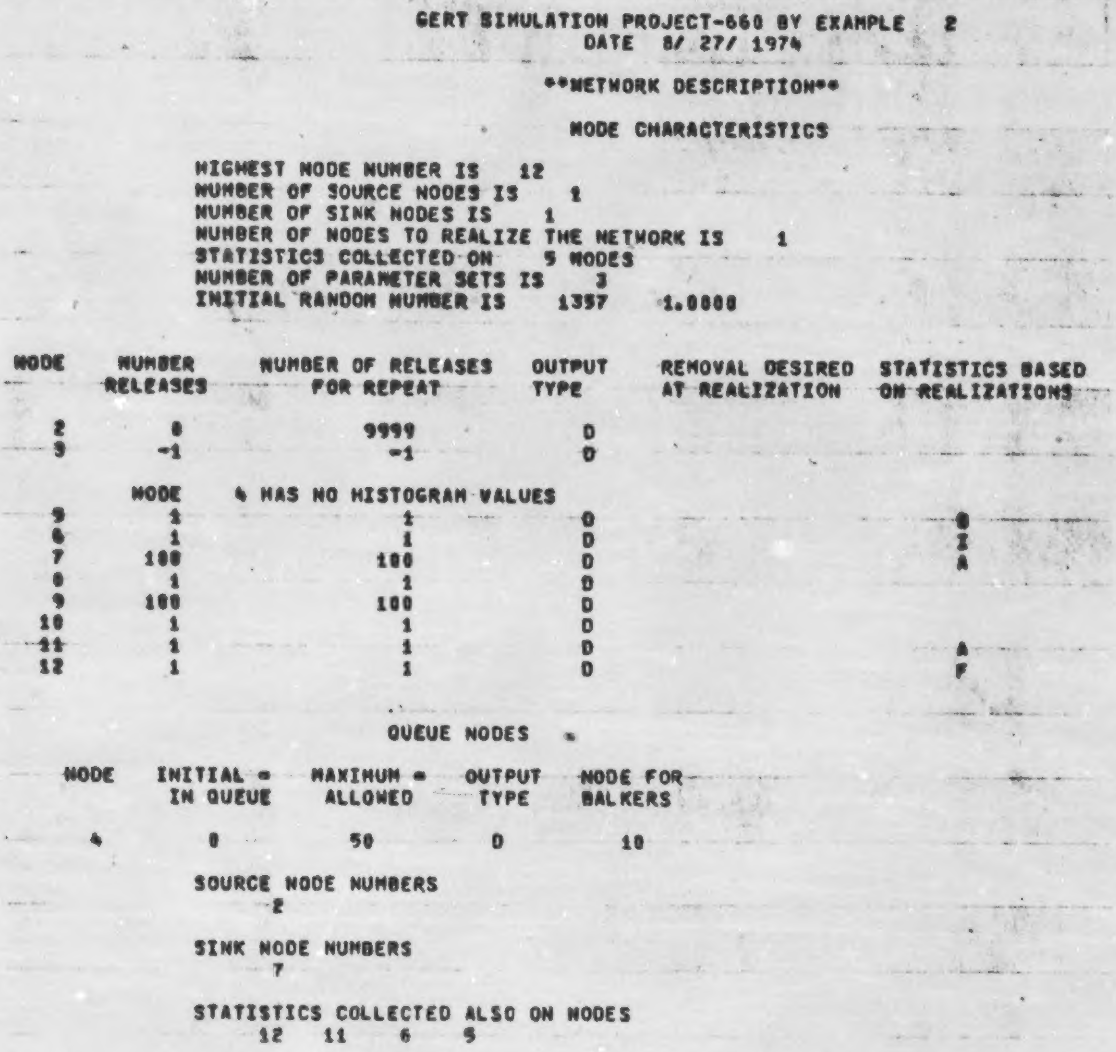


```
       EXAMPLE   2 -66008271974    5   3 48    1357      1        TYPE 1
        12  1  1  1  5     1                                      TYPE 2
         2  1                                                     TYPE 3
         3  4  1  1D                                              TYPE 3
         4  5  0 50D                  10                          TYPE 3
         5  3  1  1D    1     1  B                                TYPE 3
         6  3  1  1D    1     2  I                                TYPE 3
         7  2100100D    1    10  A                                TYPE 3
         8     1  1D                                              TYPE 3
         9   100100D                                              TYPE 3
        10     1  1D                                              TYPE 3
        11  3  1  1D    1     1  A                                TYPE 3
        12  3  1  1D    1     5  F                                TYPE 3
                                                                  TYPE 4
                  0         2.                                    TYPE 4
                  1.        3.                                    TYPE 4
           1.     2  3  1  1                                      TYPE 6
         1.       3  3  2  3                                      TYPE 6
       1.0        3  4  1  1                                      TYPE 6
       1.0        4  5  3  3                                      TYPE 6
       1.         5  6  1  1                                      TYPE 6
       1.         6  7  1  1                                      TYPE 6
       1.         3  9  1  1                                      TYPE 6
       1.         9 12  1  1     1                                TYPE 6
       1.        10 11  1  1                                      TYPE 6
       1.        11  7  1  1                                      TYPE 6
         1  3  8                                                  TYPE 7

         1  1                                                     TYPE 8
```

Figure 6.2C is a complete print-out of the computer simulation. The data input information shows that no histogram values were given for node 4. However the system can still be run since default values for the histogram are automatically assigned.

In the computer run the "people" are generated faster than the teller can handle then; the queue node (#4) begins to collect entities. However, since only 100 "people" were generated, the number of entities in the queue never exceeded 50; node #10 accordingly was never realized.

Figure 6.2C



```
                              GERT SIMULATION PROJECT-660 BY EXAMPLE   2
                                          DATE  8/ 27/ 1974

                                                  **NETWORK DESCRIPTION**

                                                  NODE CHARACTERISTICS

            HIGHEST NODE NUMBER IS   12
            NUMBER OF SOURCE NODES IS    1
            NUMBER OF SINK NODES IS    1
            NUMBER OF NODES TO REALIZE THE NETWORK IS    1
            STATISTICS COLLECTED ON    5 NODES
            NUMBER OF PARAMETER SETS IS    3
            INITIAL RANDOM NUMBER IS    1357    1.0000

NODE    NUMBER     NUMBER OF RELEASES   OUTPUT    REMOVAL DESIRED  STATISTICS BASED
       RELEASES       FOR REPEAT        TYPE     AT REALIZATION   ON REALIZATIONS

   2      0              9999             D
   3     -1                -1             D

        NODE    4 HAS NO HISTOGRAM VALUES
   5      1                 1             D                                 B
   6      1                 1             D                                 I
   7    100               100             D                                 A
   8      1                 1             D
   9    100               100             D
  10      1                 1             D
  11      1                 1             D                                 A
  12      1                 1             D                                 F

                                    QUEUE NODES  *

      NODE   INITIAL =   MAXIMUM =   OUTPUT   NODE FOR
             IN QUEUE    ALLOWED     TYPE     BALKERS

         4       0          50         D         10

            SOURCE NODE NUMBERS
                2

            SINK NODE NUMBERS
                7

            STATISTICS COLLECTED ALSO ON NODES
                12   11    6    5
```

Figure 6.2C continued

```
                         **ACTIVITY PARAMETERS**

PARAMETER                                 PARAMETERS
 NUMBER                1            2            3            4

    1                  0            0            0            0
    2                  0            0       2.0000            0
    3                  0       1.0000       3.0000            0

                         **ACTIVITY DESCRIPTION**

START      END      PARAMETER      DISTRIBUTION      COUNT      ACTIVITY      PROBABILITY
 NODE      NODE      NUMBER            TYPE           TYPE       NUMBER

   2         3         1                1               0           0            1.0000
   3         3         2                3               0           0            1.0000
   3         4         1                1              -1           0            1.0000
   3         9         1                1               0           0            1.0000
   4         5         3                3              -4           0            1.0000
   5         6         1                1               0           0            1.0000
   6         7         1                1               0           0            1.0000
   9        12         1                1               0           1            1.0000
  10        11         1                1               0           0            1.0000
  11         7         1                1               0           0            1.0000

                         **NETWORK MODIFICATIONS**

ACTIVITY NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE NODE FILE

    1      3    8


                  GERT SIMULATION PROJECT-660 BY EXAMPLE   2
                            DATE  8/ 27/ 1974


                   **FINAL RESULTS FOR    5 SIMULATIONS**

      NODE   PROB./COUNT      MEAN    STD.DEV.     = OF    MIN.       MAX.  NODE TYPE
                                                   OBS.


        7   1.0000         202.2520    5.9013        5.  195.3235  210.2836    A

       12   1.0000         103.0436    4.2593        5.   95.0745  107.0787    F
       11              0          NO VALUES RECORDED

        6   1.0000          50.0069   28.6706      500.    1.5802  106.9739    I

        5   1.0000           2.0325     .5776      500.    1.0055    3.7890    B


                            **QUEUE NODES**

   NODE     MEAN    STD.DEV.     = OF      MIN.       MAX.
                                 OBS.


     4   23.7710     1.0023        5.    20.9825    25.8766  AVERAGE NUMBER IN THE QUEUE
     4    1.0000          0        5.     1.0000     1.0000  AVERAGE BUSY TIME OF PROCESSOR
     4         0          0        5.          0          0  AVERAGE BALKERS PER UNIT TIME
```

Figure 6.2C continued

```
HISTOGRAM FOR NODE NO.    7, THE LOWER LIMIT OF THE SECOND CELL =      1.0000
THE CELL WIDTH =     10.0000

          ***FREQUENCIES***

     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      2      3
     0      0      0      0      0      0      0      0      0      0

HISTOGRAM FOR NODE NO.   12, THE LOWER LIMIT OF THE SECOND CELL =      1.0000
THE CELL WIDTH =      5.0000

          ***FREQUENCIES***

     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      1      0      3
     1      0      0      0      0      0      0      0      0      0

HISTOGRAM FOR NODE NO.   11, THE LOWER LIMIT OF THE SECOND CELL =      1.0000
THE CELL WIDTH =      1.0000

          ***FREQUENCIES***

     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      0

HISTOGRAM FOR NODE NO.    6, THE LOWER LIMIT OF THE SECOND CELL =      1.0000
THE CELL WIDTH =      2.0000

          ***FREQUENCIES***

     0     10     13     11      6      9     12     17      7      9      8
    10     11      6      6     13     15     13     16      5     11      7
    12     10      9     13      8     14      8     10     11    190
```

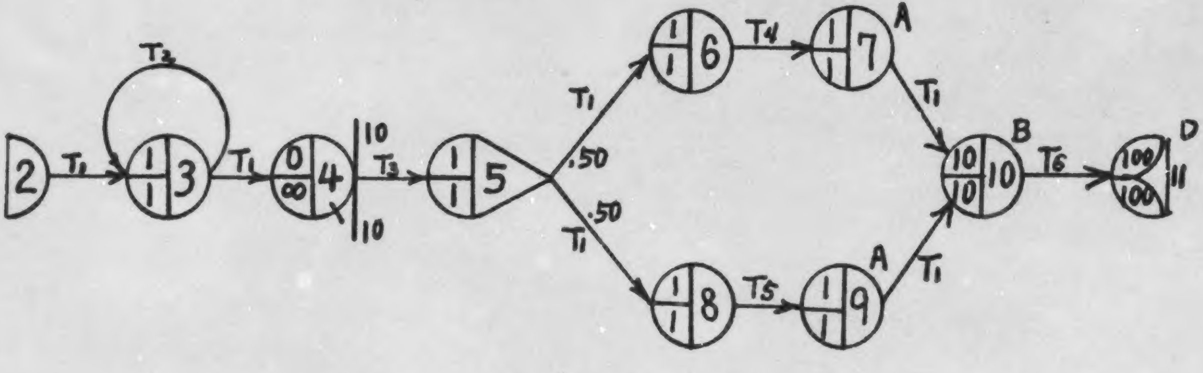
```
HISTOGRAM FOR NODE NO.    5, THE LOWER LIMIT OF THE SECOND CELL =      1.0000
THE CELL WIDTH =      1.0000

          ***FREQUENCIES***

     0    236    262      2      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      0

HISTOGRAM FOR NODE NO.    4, THE LOWER LIMIT OF THE SECOND CELL =             0
THE CELL WIDTH =      1.0000

          ***FREQUENCIES***

     0      0      0      0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0      0      0      1
     0      0      2      0      2      0      0      0      0      0
```

6.3 Wax Mystery Museum

The arrival of visitors is uniformly distributed with a minimum of 0 and a maximum of 5 minutes. There is one ticket stall, forcing all the visitors into this single waiting line. Only 10 people are allowed in the museum at a time. From the gate two paths lead to a cart which will take visitors back to the parking lot after the tour.

- capacity of cart is 10 passengers,
- time spent at the gate is between one and three minutes,
- time spent on path-1 averages 6 minutes, with a standard deviation of 2, minimum of 5 and maximum of 7,
- time spent on path-2 averages 8 minutes, with a standard deviation of 5, minimum of 6 and maximum of 10.

Figure 6.3A


2
T1
T2
1
1
3
T1
0
∞
4
10
T3
10
1
1
5
T1
.50
.50
T1
1
1
6
T4
1
1
7
A
T1
10
10
10
B
T6
100
100
D
11
1
1
8
T5
1
1
9
A
T1

Node three generates an infinite amount of people. A person is generated uniformly, with respect to time, between 0 and 5 time units. These people are sent to a waiting line (node 4) where at most, ten people can go on the tour. There is a processing time for the ticket collection (the time is uniformly distributed, 1-3). Once the tourist enters, he decides on which tour to go. Each tour has an equal chance of being chosen. One tour's time (branches 5-6-7-10) is also normally distributed with a mean of 6, standard deviation of .2, minimum of 5, and maximum 7. The second tour (branches 5-8-9-10) is also normally distributed with mean 8, standard deviation of .5, minimum 6, and maximum 10. When an entity reaches node 10, a signal is sent to gated queue node 4 to let one more person into the tour. The time on branch 10-11 represents the travel time for the cart to take the people back to the parking lot. Statistics are collected on nodes 7 and 9, because it is essential to know how many people went on each tour. Node 10 collects the time between realizations. This is useful to know because we can then tell how many carts would be needed to avoid a bottleneck.

Figure 6.3B shows a listing of the data cards for the simulation run.

Figure 6.3B

```
EXAMPLE;  3  66008271974  10   6  44    1357               TYPE   1
 11  1  1  1  4        1                                   TYPE   2
  2  1                                                     TYPE   3
  3     1  1D                                              TYPE   3
  4  5  0  0D     1     1  A                               TYPE   3
  5     1  1P                                              TYPE   3
  6     1  1D                                              TYPE   3
  7  3  1  1D     1     1  A                               TYPE   3
  8     1  1D                                              TYPE   3
  9  3  1  1D     1     1  A                               TYPE   3
 10  3 10 10D     1     1  B                               TYPE   3
 11  2100100D     1     1  D                               TYPE   3

                                                           TYPE   4
                0         5.                               TYPE   4
                1.        3.                               TYPE   4
     6.         5.        7.         .2                    TYPE   4
     8.         6.       10.         .5                    TYPE   4
     2.                                                    TYPE   4
   4 10 10                                                 TYPE   5
    1.00  2  3  1  1                                       TYPE   6
    1.00  3  3  2  3                                       TYPE   6
    1.00  3  4  1  1                                       TYPE   6
    1.00  4  5  3  3                                       TYPE   6
    0.50  5  6  1  1                                       TYPE   6
    0.50  5  8  1  1                                       TYPE   6
    1.00  6  7  4  2                                       TYPE   6
    1.00  7 10  1  1                                       TYPE   6
    1.00  8  9  5  2                                       TYPE   6
    1.00  9 10  1  1                                       TYPE   6
    1.00 10 11  6  1                                       TYPE   6


   1  1                                                    TYPE   8
```

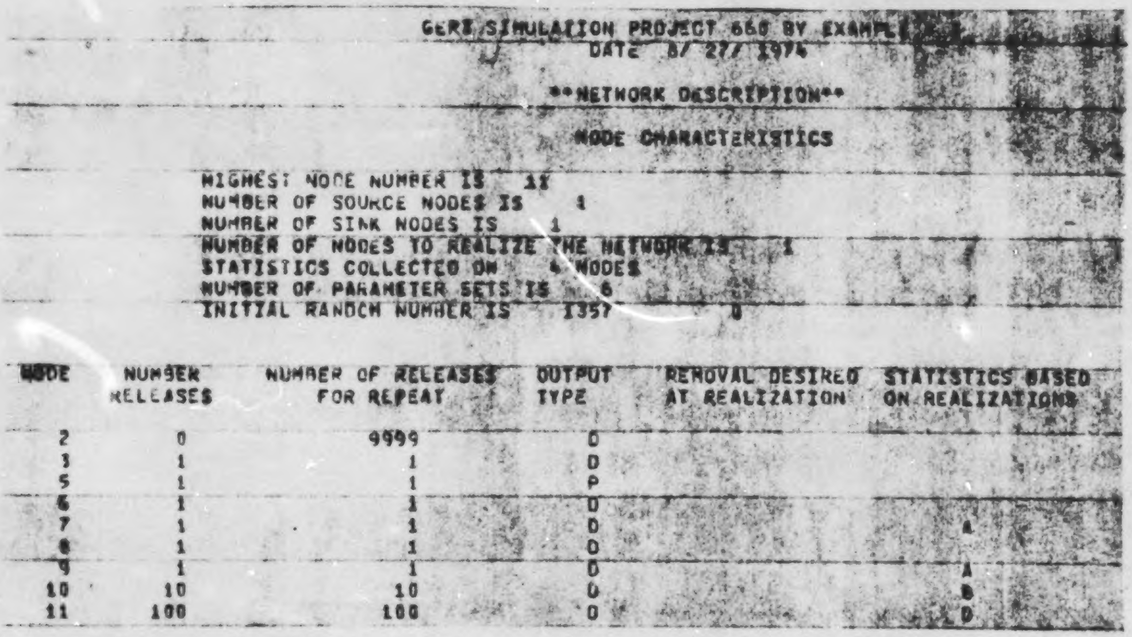
The results from the simulation can be interpreted as follows:

- The queue statistics show there is no waiting line. This indicates there was always less than or equal to ten people on both tours. As stated before, nodes

7 and 9 are there to tell how many people went on each tour (Note: each tour got roughly half the people, as was requested on data card type 5)

- Node 10 shows that for every 25 time units (average), a cart is filled with people and leaves. Since the cart takes 2 time units to make the round trip, there is no need for additional carts; 23 time units out of 25 the cart is free.
- For this example, a delay node has been used, node 11 (the sink node). This shows that on the average, 2470 time units passed between the first and last realization .
- Some of the histograms do not tell as much as possible; the cell parameters are not correctly set. For nodes 7 and 9, the limit should be 5.0 and the cell width 87.0. After a program has been run, the limits should be checked and reset to give more information.

Figure 6.3C



```
                              GERT SIMULATION PROJECT 660 BY EXAMPLE
                                        DATE  8/ 27/ 1976

                                      **NETWORK DESCRIPTION**

                                       NODE CHARACTERISTICS

              HIGHEST NODE NUMBER IS   11
              NUMBER OF SOURCE NODES IS     1
              NUMBER OF SINK NODES IS     1
              NUMBER OF NODES TO REALIZE THE NETWORK IS    1
              STATISTICS COLLECTED ON    4 NODES
              NUMBER OF PARAMETER SETS IS    6
              INITIAL RANDOM NUMBER IS    1357          0

NODE     NUMBER      NUMBER OF RELEASES   OUTPUT    REMOVAL DESIRED   STATISTICS BASED
        RELEASES         FOR REPEAT        TYPE     AT REALIZATION    ON REALIZATIONS

  2        0                9999             D
  3        1                   1             D
  5        1                   1             P
  6        1                   1             D
  7        1                   1             D                               A
  8        1                   1             D
  9        1                   1             D                               A
 10       10                  10             D                               B
 11      100                 100             D                               D
```

Figure 6.3C continued

```
                         QUEUE NODES

   NODE    INITIAL #   MAXIMUM #   OUTPUT   NODE FOR
           IN QUEUE    ALLOWED     TYPE     BALKERS

     4        0         9999         0         0

                SOURCE NODE NUMBERS
                      2

                SINK NODE NUMBERS
                     11

                STATISTICS COLLECTED ALSO ON NODES
                     10    9    7


                        **ACTIVITY PARAMETERS**

  PARAMETER                                PARAMETERS
   NUMBER                1                2               3              4

      1                  0                0               0              0
      2                  0                0          5.0000              0
      3                  0           1.0000          3.0000              0
      4             6.0000           5.0000          7.0000          .2000
      5             8.0000           6.0000         10.0000          .5000
      6             2.0000                0               0              0

                          **GATED QUEUES**

  QUEUE              SPECIAL              NUMBER
  NODE               NODE                 LEAVING
      4                  10                 10

                       **ACTIVITY DESCRIPTION**

  START     END     PARAMETER     DISTRIBUTION     COUNT     ACTIVITY     PROBABILITY
  NODE      NODE    NUMBER           TYPE          TYPE      NUMBER

    2         3        1               1             0          0            1.0000
    3         3        2               3             0          0            1.0000
    3         4        1               1            +1          0            1.0000
    4         5        3               3             0          0            1.0000
    5         6        1               1             0          0             .5000
    5         8        1               1             0          0             .5000
    6         7        4               2             0          0            1.0000
    7        10        1               1            -4          0            1.0000
    8         9        5               2             0          0            1.0000
    9        10        1               1            -4          0            1.0000
   10        11        6               1          -271          0            1.0000

                GERT SIMULATION PROJECT 660 BY EXAMPLE 1 3
                            DATE  8/ 27/ 1974


                **FINAL RESULTS FOR   10 SIMULATIONS**

   NODE   PROB./COUNT      MEAN    STD.DEV.    # OF     MIN.       MAX.   NODE TYPE
                                               OBS.

    11    1.0000     2469.5240   51.1995       10.  2418.0296  2585.4474     D

    10    1.0000       25.0368    4.9341     1000.     9.1424    39.7608     B

     9    1.0000     1263.6392  719.1110     5042.     8.9851  2615.5318     A

     7    [illegible]     1254.8742  [illegible]     [illegible]     [illegible]  [illegible]     A
```

Figure 6.3C continued

```
                                **QUEUE NODES**

        NODE     MEAN     STD.DEV.      # OF         MIN.        MAX.
                                        OBS.

         4      .0000      .0000        10.            0       .0001  AVERAGE NUMBER IN THE QUEUE
         4          0          0        10.            0           0  AVERAGE BALKERS PER UNIT TIME

                 HISTOGRAM FOR NODE NO.    11, THE LOWER LIMIT OF THE SECOND CELL =      1.0000

                 THE CELL WIDTH =      1.0000

                         ***FREQUENCIES***

                    0     0     0     0     0     0     0     0     0     0     0
                    0     0     0     0     0     0     0     0     0     0     0
                    0     0     0     0     0     0     0     0     0    10

                 HISTOGRAM FOR NODE NO.    10, THE LOWER LIMIT OF THE SECOND CELL =      1.0000

                 THE CELL WIDTH =      1.0000

                         ***FREQUENCIES***

                    0     0     0     0     0     0     0     0     0     2     1
                    0     2     2     9    12    19    28    36    40    50    66
                   76    85    85    75    55    67    67    59    43   115

                 HISTOGRAM FOR NODE NO.     9, THE LOWER LIMIT OF THE SECOND CELL =      1.0000

                 THE CELL WIDTH =      1.0000

                         ***FREQUENCIES***

                    0     0     0     0     0     0     0     0     1     1     1
                    1     2     1     5     2     1     3     2     1     1     0
                    3     1     1     2     3     2     2     2     2  5002

                 HISTOGRAM FOR NODE NO.     7, THE LOWER LIMIT OF THE SECOND CELL =      1.0000

                 THE CELL WIDTH =      1.0000

                         ***FREQUENCIES***

                    0     0     0     0     0     0     1     4     1     3     4
                    2     2     0     1     3     2     3     3     3     2     2
                    2     1     0     1     2     2     0     1     3  4918

                 HISTOGRAM FOR NODE NO.     4, THE LOWER LIMIT OF THE SECOND CELL =      1.0000



                 THE CELL WIDTH =      1.0000

                         ***FREQUENCIES***

                   10     0     0     0     0     0     0     0     0     0     0
                    0     0     0     0     0     0     0     0     0     0     0
                    0     0     0     0     0     0     0     0     0     0
```

GERTS GQ Simulator for Systems Simulation

Part II: Programmer's Manual

1. Scheduling

GASP is the master control routine. GASP starts the simulation, selects events, sequences time, controls the monitoring of intermediate simulation results, and initiates the print out of the final output when the simulation has been completed. NSET is the main array which stores the scheduled events. While it is a one-dimensional array, it can be thought of as two dimensional with seven columns. The first five columns contain:

- end node number
- time parameter number
- time distribution number
- special coding concerning queues or delay nodes
- network modification number

The last two columns are successor and predecessor pointer columns. The number 7777 indicates no predecessor, 9999 indicates no successor. The first "N" rows contain the original data from card type 6 (where N represents how many branches are input), after that the other interacting subroutines add or delete new rows as necessary.

2. Subroutine Descriptions

The program is divided into three parts for overlay purposes so that there are three program names:

- PROGRAM GERTSGQ for those parts that remain in core the entire time,
- PROGRAM DATAN for those parts that are in the first overlay, and

- PROGRAM GASP for those parts that are in the second overlay.

2.1 In Core Routines

PROGRAM GERTSGQ: This program initializes the card reader, printer, IMN, IMNN, and calls SUBROUTINE DCNBGD.

SUBROUTINE DCNBGD: The first overlay is called; if no input errors were recognized, it calls the second overlay, then returns to PROGRAM GERTSGQ.

FUNCTION DRAND(IY): This is the random number generator.

SUBROUTINE ERROR: An error message is printed, followed by NSET and QSET, the major arrays used to hold the event information.

SUBROUTINE EXIT: Exits from the program by calling exit macro.

SUBROUTINE FILEM: Future events, with their attributes are stored in NSET, with pointers directing the sequence. Sorting according to time is also done.

2.2 Overlay 1

PROGRAM DATAN: Reads data card type 1 and 2. These variables initialize the capacities of the system. Data card type 3 is read and arrays for statistics, characteristics of the nodes, and queues are initialized. Data cards type 4 and 5 are read, the information is stored in arrays and the table of activities along with their pointers are set equal to zero to prepare for the next set of cards, data card type 6. This type of card contains branch information. Some time parameters are changed depending on the distribution. Branches are defined which are connected to queue nodes, delay nodes, or the end node for a gated queue. Files information concerning branches into table of activities. Additional statistic node activity and histogram values are initialized. Data card type 7 is read and network modification arrays are initialized. After reading data card type 8, a test is made to determine if there is an error in the data input; if so, the second overlay is not called.

SUBROUTINE BETAXF: The subroutine converts the Beta parameters (mean, variance) to a mean and variance over (0,1) interval, and calculates values needed to obtain a Beta variate.

FUNCTION PERTXF: Fits time parameters into beta distribution.

2.3 Overlay 2

PROGRAM GASP: Items necessary for activity profile are initialized, times are set to 0, queue node pointers are reinitialized, etc. Simulation begins. Source node items are scheduled. Each time an event occurs tests for the following items are made and appropriate action taken:

- trace or activity profile
- network modification
- counter-type specification
- delay node
- entering or leaving a queue node
    - update queue information
    - if gate is open and entities are in queue, schedule another departure
    - check for too many in queue, divert to balk node

if necessary

  - if balk node is utilized, file event in NSET
- test for mark node
- test for statistics gathering
  - gather appropriate statistic
- test for end of run
- test for end of simulation run.

FUNCTION BETA(J): Random numbers with a Beta distribution are calculated. PGAMMA function is called.

SUBROUTINE COLCT: Statistics are collected which include the sum, sum of squares, number of observations, minimum and maximum.

FUNCTION ERLANG: Random number with Erlang distribution is obtained.

FUNCTION NFIND: Used if removal has been specified. Selectively eliminates scheduled events.

FUNCTION GAMMA5: This routine calculates a gamma variate.

SUBROUTINE HISTO: This subroutine is called when one of the cells in the histogram of a statistics node is to be incremented.

FUNCTION RLOGN: Calculates the normal distribution, then sets it as a power of the exponential.

SUBROUTINE NPOSN: Calculates time from a function of the poisson distribution.

FUNCTION PGAMMA: Calculates a gamma variate needed in the calculation of beta time distribution.

SUBROUTINE RMOVE: Puts the attributes of the branch into JTRIB and ATRIB. Sets up pointers in NSET.

FUNCTION RNORM: Returns a time calculated from a normal distribution. Randomly calculates the variation (+ or -) from the mean and then adds to the mean.

SUBROUTINE SAMPLE: Using the proper distribution, the arrival time for next node is calculated.

SUBROUTINE SCHAT: Schedules future events which result from the realization of a node. First it determines if a node is probabilistic or deterministic. If probabilistic, it tests which branch to use according to a random number. When the proper branch has been selected, the event is scheduled and filed.

FUNCTION UNFRM: Calculates a random number fitted to the uniform distribution.

SUBROUTINE SUMRY: Writes the results of simulation. Calculates averages, and standard deviations. Statistics for statistic nodes, counter types, and queue nodes are written, followed by frequency tables.

3. Common Variables

ATRIB(2):	ATRIB(1): probability of a branch ATRIB(2): is used for marking
ID:	number of simultaneous activities
IM:	set to 5 in DATAN
IMM:	set to 2 in DATAN
IMN:	set to 270 in GERTS GQ
IMNN:	set to 325 in GERTS GQ
IMNQ(50):	an array containing the node numbers for activity profile
ISEED:	seed for random number generator

JCELS(99,32):	frequencies for histograms
JSINK(100):	each element is 1 through 5 depending on type of statistic to be collected
JTRIB(5):	event code; 1) end node, 2) parameter #, 3) distribution type, 4) countertype, 5) modification number
KOUNT(5):	statistics for count types kept in each element
MAXNS;	ID*MXX = # of simultaneous activities
MAXQS:	ID*IMM = # of simultaneous activities X 2
MFA:	points to node from which next branches are to be scheduled
MFAQ:	a pointer for the entities inside queue
MFE(175):	relative address of first entry of file 1 points to next node to occur
MFEQ(100):	performs for the queue the same function as MFE
MLE(175):	relative address of last entry of file 1 points to last node that occurred
MLEQ(100):	performs for the queue the same function as MLE
MON:	month
MSTOP:	set to 0 in DATAN
MX:	set to IM + 1 = 6 in DATAN
MXC:	set to 32 in DATAN; number of cells in histogram
MXX:	set to IM + 2 = 7 in DATAN
NABA(270):	array to store network mod#; NABA(270)

	and NABA(269) stores run numbers to be traced
NAME(6):	name of program (6A2)
NCLCT:	(# of stat nodes) X (# of count types + 1) + 3X(# of Q nodes)
NCRDR:	dsi of card reader
NCTS:	# of types of counts
NDAY:	day of the month
NDT:	# of delay type nodes (stat node with D-code)
NDTL:	number of sink and delay nodes + 1
NDTU:	sum of number of sink, delay, # of delay type nodes
NFTBU(175):	a holder of node numbers; used for network modification
NHIST:	number of stat nodes + # of Q-nodes
NN:	highest node number
NNM:	= 1 if network mod exists, 0 if not
NOQ:	highest node #; set equal to NN in DATAN
NPD:	number of sink nodes
NPO(60):	set of pointers to NABA for network mod
NPRMS:	number of different time parameter cards
NPRNT:	dsi of printer
NPROJ:	project # (from data card 1); if less than 0, get trace
NPTR(325):	a pointer for QATT
NQ(175):	contains number of branches which each node emits
NQL:	1 more than the number of stat nodes

NQN:	# of queue nodes
NQU:	the number of Q nodes + stat nodes
NREL(175):	counter of how many entities are in a node
NREL2(175):	number of releases for each node after first time
NRELP(175):	number of releases for each node for first time
NRUN:	counter of how many runs have been made
NRUNS:	number of simulation runs to be performed
NSINK(100):	an array of stat node numbers
NSKD:	sink node and delay node
NSKS:	# of nodes for statistics to be collected
NSKSR:	determiner when system is to be stopped, counts how many sink nodes have been realized
NSKST:	# of sink nodes needed for realization of system
NSNR(100):	number of sink/stat realizations
NSORC(1):	the node number of the source node
NSRC:	# of source nodes
NSTND:	# of stat nodes but not a delay node
NSTUS(100):	contains status of processor for a node
NTYPE(175):	for each node 1 if deterministic and not removed, 2 if prob and not remove, 3 if remove and determ., 4 if remove and prob.
NYR:	year
PARAM(100,4):	time parameters
QATT(325):	contains times in queue which the mark node

	gives each entity
SBUS(100):	sum of the busy time for each processor
SEED:	signal for activity profile
SQUE(100):	sum of number in queue integrated over time
SUMA(99,5):	(I,1) sum to calculate mean, (I,2) square to get standard deviation, (I,3) # of obs., (I,4) min., (I,5) max.
TLXC(100):	time of last change for the queue or processor
TNOW:	initialized to 0; the time sense
TOTIM:	total time
WLDTH(99):	width of cell for histogram
XBALK(33):	sum of the number of balkers
XLOW(99):	array to hold XLL, the lower bound of second cell of histogram
XSTUS(99):	saves time for each statistic or queue node

4. Program Listing

4.1 In Core Routines

```
      PROGRAM GERTS3GQ
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMON NDT,NDTL,NDTU,NSKD,NSTND,XSTUS(99)
      COMMON NON,NQL,NQU,NSTUS(100),SBUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEQ(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
      COMMON NSET(2275),QSET(1350)
C*****     MAIN PROGRAM FOR GERTS IIIQ
C
C*****SET EQUIPMENT NUMBERS FOR CARD READER(NCRDR) AND PRINTER (NPRNT).
C
      NCRDR=60
      NPRNT=61
      IMN=270
      IMNN=325
   10 CALL DCNBGD
      GO TO 10
      END
```

```
      SUBROUTINE DCNBGD
C     THIS SUBROUTINE IS SET UP BY
C     DO HUU CHUYEN
C     NANCY BLOM
C     GARY DURAN
C     FOR OVERLAY PURPOSE
C
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMON NDT,NDTL,NDTU,NSKD,NSTND,XSTUS(99)
      COMMON NON,NQL,NQU,NSTUS(100),SBUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEQ(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
      COMMON IGATE(50,2),NUMGQ,ITEST(50)
      COMMON NSET(2275),QSET(1350)
      CALL OVERLAY(1,0,55)
      IF(NRUNS.LT.0)RETURN
      CALL OVERLAY(2,0,55)
      RETURN
      END
      FUNCTION DRAND(IY)
      IY=IY*(2**12+3)
      IF(IY) 5,6,6
5     IY=IY+8388607
6     YFL=IY
      DRAND=YFL*.1192093E-6
      RETURN
      END
```

```
      SUBROUTINE ERROR(J,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      WRITE(NPRNT,99)
      WRITE (NPRNT,100) J,TNOW
  100 FORMAT(//36X16HERROR EXIT, TYPE,I3,7H ERROR.//21H FILE STATUS AT
     1TIME,F10.4/)
      WRITE (NPRNT,200)
  200 FORMAT(30X,4HNSET/)
      DO 210 I=1,ID
      IL=(I-1)*MXX+1
      IV=IL+MXX-1
  210 WRITE(NPRNT,90) I,(NSET(IJ),IJ=IL,IV)
   90 FORMAT(13X,I5,5X,12I8)
      WRITE (NPRNT,202)
  202 FORMAT(//30X,4HQSET/)
      DO 215 I=1,ID
      IL=(I-1)*IMM+1
      IV=IL+IMM-1
  215 WRITE(NPRNT,95) I,(QSET(IJ),IJ=IL,IV)
   95 FORMAT(13X,I5,4X,10(E12.6,2X))
      WRITE (NPRNT,99)
   99 FORMAT(1H1)
      IF(NCLCT)9,9,8
    8 WRITE (NPRNT,98)
   98 FORMAT(/11H ARRAY SUMA,/)
      DO 110 I=1,NCLCT
  110 WRITE (NPRNT,80) I,(SUMA(I,K),K=1,5)
   80 FORMAT(I10,5F10.4)
      WRITE (NPRNT,99)
    9 IF(NHIST) 11,11,12
   12 WRITE (NPRNT,96)
   96 FORMAT (/12H ARRAY JCELS/)
      NCL=MXC
      DO 112 I=1,NHIST
  112 WRITE (NPRNT,26) I,(JCELS(I,K),K=1,NCL)
   26 FORMAT(17X,I3,5X,11I4/(25X,11I4))
   11 NRUNS=-4
      RETURN
      END
      SUBROUTINE FILEM (JQ,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,MCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
C
C*****TEST TO SEE IF THERE IS AN AVAILABLE COLUMN FOR STORAGE
C
      IF (MFA - ID ) 2,2,3
    3 WRITE (NPRNT,4)
    4 FORMAT (//24H OVERLAP SET GIVEN BELOW/)
      CALL ERROR (87,NSET,QSET)
      RETURN
C
C*****PUT ATTRIBUTE VALUES IN FILE
C
    2 INDX = (MFA - 1) * IMM
      DO 1 I = 1,IMM
      INDX = INDX + 1
    1 QSET(INDX) = ATRIB(I)
      INDX = (MFA - 1) * MXX
      DO 10 I = 1,IM
      INDX = INDX + 1
   10 NSET(INDX) = JTRIB(I)
      MFEX=MFE(JQ)
      KNT = 2
      KS=1
      INDX=MFA*MXX-1
      NXFA = NSET(INDX)
      MLEX=MLE(JQ)
      IF(MLEX)100,27,11
  100 CALL ERROR (88,NSET,QSET)
      RETURN
   27 INDX=MFA*MXX
      NSET(INDX) =9999
      MFE(JQ) = MFA
   17 INDX = MFA * MXX - 1
      NSET(INDX) =7777
      MLE(JQ) = MFA
   14 MFA =NXFA
      IF(MFA-8888)237,238,238
  237 INDX=NXFA*MXX
      NSET(INDX) =9999
  238 NQ(JQ)=NQ(JQ)+1
      RETURN
   11 INDX1=(MFA-1)*IMM+KS
      INDX2 = (MLEX - 1) * IMM + KS
      IF(JQ-1) 241,241,242
  241 IF (QSET(INDX1) - QSET(INDX2)) 12,13,13
  242 IF (QSET(INDX1) - QSET(INDX2)) 13,13,12
   13 INDX = MLEX * MXX - 1
      MSU = NSET(INDX)
      NSET(INDX) = MFA
   INDX = MFA * MXX
   NSET(INDX) = MLEX
   GO TO (18,17),KNT
18 INDX = MFA * MXX - 1
   NSET(INDX) = MSU
   INDX = MSU * MXX
   NSET(INDX) = MFA
   GO TO 14
12 KNT = 1
   INDX = MLEX * MXX
   MLEX = NSET(INDX)
   IF(MLEX-9999) 11,16,11
16 INDX = MFA * MXX
   NSET(INDX) =9999
   MFE(JQ) = MFA
   INDX = MFA * MXX - 1
   NSET(INDX) = MFEX
   INDX = MFEX * MXX
   NSET(INDX) = MFA
   GO TO 14
   END
```

```
    SUBROUTINE EXIT
    CALL Q8QEXITS
    RETURN
    END
```

4.2 Overlay 1

```
      PROGRAM DATAN
      DIMENSION NABYA(22),MDATA(5)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMON NDT,NDTL,NDTU,NSKD,NSTND,XSTUS(99)
      COMMON NQN,NQL,NQU,NSTUS(100),SBUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEQ(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
      COMMON IGATE(50,2),NUMGQ,ITEST(50)
      COMMON NSET(2275),QSET(1350)
      DIMENSION IPP(2),IRR(2),ICD(2)
      DIMENSIONNA(1),NF(1),NB(1),NI(1),ND(1),NLIFO(1),NFIFO(1)
      EQUIVALENCE(PP,IPP),(DD,IDD),(RR,IRR)
      DATA (IPP=1HP,1H ),(IRR=1HR,1H ),(IDD=1HD,1H ),(NA=1HA),(NF=1HF),
     1(NI=1HI),(ND=1HD),(NB=1HB)
      MDATA(1)=NF
      MDATA(2)=NA
      MDATA(3)=NB
      MDATA(4)=NI
      MDATA(5)=ND
      NRUN = 1
      READ(NCRDR,101) NAME,NPROJ,MON,NDAY,NYR,NRUNS,NPRMS,ID,JSEED,SEED
  101 FORMAT(6A2,I4,2I2,4I4,I8,F10.4)
      IF(NRUNS) 30,30,20
   30 CALL EXIT
   20 WRITE(NPRNT,21)
   10 FORMAT(25I3)
  444 FORMAT(4I3,2A1,2F6.2,A1,A3,I3)
      K1=1
C
C*****READ LARGEST NODE NUMBER, NO. OF SINK NODES, NO. OF SOURCE NODES,
C*****NO. OF NODES NECESSARY TO REALIZE THE NETWORK,NODES MOD.,PRT CDE
C*****NOTE:  NO NOT INCLUDE THE NUMBER OF QUEUE NODES IN NSKS
C*****HOWEVER DO INCLUDE THE NUMBER OF D NODES IN NSKS
C
      WRITE(NPRNT,22) NPROJ,NAME,MON,NDAY,NYR
C $$$ NUMGQ = NUMBER OF GATED QUEUES.  FIRST COLUMN OF IGATE HAS
C $$$ QUEUE NUMBER, 2ND COLUMN HAS NODE  WHICH TURNS Q BACK ON.
C $$$ ITEST TELLS HOW MANY TO LET THROUGH BEFORE GATING.
      READ(NCRDR,10) NN,NSRC,NPD,NSKST,NSKS,NCTS,NNM,NUMGQ
      NSKSR=NSKST
      WRITE(NPRNT,221)NN,NSRC,NPD ,NSKST,NSKS,NPRMS,JSEED,SEED
      DO 707 I=1,NN
  707 NTYPE(I)=0
      DO 1707 L=1,NSKS
 1707 XSTUS(L)=-1.0
      II=0
      NSKND=0
      NSKD=NPD
      NSTD=NPD
      NSTND=NSKS
C
C*****NSKND--SINK NODE AND NOT A DELAY NODE
C*****NSKD--SINK AND DELAY NODE
C*****NSTD--STAT AND DELAY NODE
C*****NSTND--STAT AND NOT A DELAY NODE
C
      MM=NSKS
      NDT=0
C*****NDT--NUMBER OF DELAY TYPE NODES
      NQN=0
C*****NQN--THE NUMBER OF QUEUE NODES
      WRITE(NPRNT,229)
      DO 410 I=1,NN
      NRR=0
C
C*****NOTE:  COUNTER TYPES ARE NOT ALLOWED FOR ACTIVITIES INTO A ≠Q≠ NOD
C*****NOTE:  A DELAY NODE CANNOT BE A QUEUE NODE; VICE VERSA
C
      READ(NCRDR,444) NODE,INOUT,MRELP,MREL2,PTOPT,PRMV,XLL,WIDH,MSINK,I
     1NT,NIFBK
C
C     NIFBK -- NODE TO WHICH WE GO IF IT IS A Q NODE AND WE HAVE BALKERS
C    IF INT = 0, FIFO       IF INT = 1, LIFO
C*****NOTE:  FOR A C-NODE WE REQUIRE A LOWER LIMIT AND WIDTH FOR THE
C***** AVERAGE NUMBER IN THE QUEUE
C*****NOTE--THE LAST CARD OF THIS TYPE MUST HAVE ≠0≠ IN COLUMN 3
C****INOUT CODE  1=SOURCE, 2=SINK, 3=STAT, 4=MARK,5=QUEUE,0=OTHERWISE
C      MSINK    JSINK                   DESCRIPTION
C        F        1      STAT NODE FOR 1ST REALIZATIONS
C        A        2      STAT NODE FOR ALL REALIZATIONS
C        B        3      STAT NODE FOR BETWEEN REALIZATIONS
C        I        4      STAT NODE FOR INCREMENT NODE
C        D        5      STAT NODE FOR DELAY NODE REALIZATIONS
C*****FOR ACTIVITIES COMING INTO A D-NODE, WE CANNOT HAVE COUNTER TYPES
C
      IF(NODE) 433,433,434
  434 IF(SEED.EQ.0.0.OR.NPROJ.GE.0) GO TO 435
      IF(INT.EQ.3H NO.OR.INT.EQ.3HNO  ) GO TO 435
      IF(K1.LT.50)GO TO 436
      WRITE(NPRNT,8000)
 8000 FORMAT(20H  ERROR IN TSD INPUT )
      NRUNS=-1
      GO TO 435
C
C***** IMNQ IS THE ARRAY OF NODE NO. FOR THE TSD
C
  436 IMNQ(K1)=NODE
      K1=K1+1
  435 NRELP(NODE)=MRELP
      NREL2(NODE)=MREL2
      IF(NREL2(NODE))1471,471,470
 1471 NREL2(NODE)=0
```

```
      GO TO 470
  471 NREL2(NODE)=9999
  470 NREL(NODE)=NRELP(NODE)
      IF(INOUT) 411,404,411
  411 GO TO (401,680,680,1644,625),INOUT
  680 DO 669 INPT=1,5
      IF (MSINK.EQ.MDATA(INPT)) GO TO 670
  669 CONTINUE
      WRITE(NPRNT,1496) NODE
 1496 FORMAT(1X4HNODE,I5,24H NEEDS A STATISTICS CODE)
      INPT=1
  670 IF(INOUT-2) 402,402,403
 1644 NREL(NODE)=-1
      NRELP(NODE)=-1
      NREL2(NODE)=-1
      GO TO 404
  401 II=II+1
      NSORC(II)=NODE
      IF(II-NSRC) 404,404,1495
 1495 WRITE(NPRNT,496) NODE
      GO TO 404
  402 IF(INPT-4) 621,621,622
  622 KKK=NSKD
      NSKD=NSKD-1
  505 NDT=NDT+1
C
C*****XSTUS--STATUS OF A ≠D≠ NODE (DELAY NODE);  IF = -1,D-NODE HAS NOT
C*****  REALIZED THE FIRST TIME   IF POSITIVE, IT IS THE TIME THE D-NODE
C*****  REALIZED THE FIRST TIME.
C
  502 JSINK(KKK)=INPT
      NSNR(KKK)=0
C*****NSNR IS DEFINED AS THE NUMBER OF SINK/STAT NODE REALIZATIONS
      GO TO 630
  621 NSKND=NSKND+1
      KKK=NSKND
      IF(NSKND-NPD) 504,504,495
  504 IF(INPT-1) 501,501,502
  501 JSINK(KKK)=1
      GO TO 630
  403 IF(INPT-4) 623,623,624
  624 NSTD=NSTD+1
      KKK=NSTD
      GO TO 505
  623 KKK=NSTND
      NSTND=NSTND-1
      IF(NSTND-NPD) 495,506,506
  506 IF(INPT-1) 501,501,502
  495 WRITE(NPRNT,496) NODE
  496 FORMAT(/5X,4HNODE,I4,61H CAUSES NO. OF STAT / SINK NODES TO EXCEED
     1 THE NO. SPECIFIED.)
      GO TO 404
  625 MM=MM+1
      KKK=MM
      JSINK(KKK)=NIFBK
      NQN=NQN+1
C*****NSTUS--STATUS OF THE PROCESSOR
      NSTUS(NQN)=0
C*****TLC--TIME LAST CHANGE FOR THE QUEUE OR PROCESSOR
      TLC(NQN)=0.0
C*****SQUE--SUM OF THE NUMBER IN THE QUEUE INTEGRATED OVER TIME
      SQUE(NQN)=0.0
C*****SBUS--SUM OF THE BUSY TIME FOR THE PROCESSOR
      SBUS(NQN)=0.0
C*****XBALK--SUM OF THE NUMBER OF BALKERS
      XBALK(NQN)=0.0
  630 NSINK(KKK)=NODE
      XLOW(KKK)=XLL
      WIDTH(KKK)=WIDH
      IF(WIDTH(KKK)) 404,429,404
  429 WIDTH(KKK)=1.0
      ATRIB(2)=0.0
      WRITE(NPRNT,431) NODE
  431 FORMAT(/5X,9H     NODE,I5,24H HAS NO HISTOGRAM VALUES)
  404 IF(PRMV-RR) 406,405,406
  405 NRR=1
  406 IF(PTOPT-PP) 407,408,407
  407 NTOPT=1
      PTOPT=DD
      GO TO 409
  408 NTOPT=2
  409 NTYPE(NODE)=2*NRR + NTOPT
      IF(INOUT.EQ.5) GO TO 410
      WRITE(NPRNT,233) NODE,NRELP(NODE),NREL2(NODE),PTOPT,PRMV,MSINK
  410 CONTINUE
  433 IF(NQN.EQ.0) GO TO 3433
      WRITE(NPRNT,523)
      NQL=NSKS+1
      NQU=NSKS+NQN
      DO 521 I=NQL,NQU
      NODE=NSINK(I)
      PTOPT=DD
      IF (NTYPE(NODE).EQ.2)PTOPT=PP
      IJ=I-NSKS
      IKODE=NFIFO
      IF(IMNQ(IJ).EQ.1) IKODE=NLIFO
  523 FORMAT(//30X,11HQUEUE NODES//5X,7HNODE     42HINITIAL =    MAXIMUM =
     1   OUTPUT   NODE FOR  /13X,40HIN QUEUE     ALLOWED      TYPE     BALK
     ZERS  /)
  524 FORMAT(I8,I9,I12,9X,A1,I11)
  521 WRITE(NPRNT,524)NODE,NRELP(NODE),NREL2(NODE),PTOPT,JSINK(I)
 3433 WRITE(NPRNT,223) (NSORC(J),J=1,NSRC)
      WRITE(NPRNT,225) (NSINK(K),K=1,NPD)
      K1=K1-1
C
C*******THIS SECTION SORTS THE TSD NODE NUMBERS, USED LATER IN GASP
C
 8001 M=0
```

```
      DO 8002 I=2,K1
      IF(IMNQ(I).GE.IMNQ(I-1))GO TO 8002
      IGARBAGE=IMNQ(I-1)
      IMNQ(I-1)=IMNQ(I)
      IMNQ(I)=IGARBAGE
      M=1
 8002 CONTINUE
      IF(M.GT.0)GO TO 8001
      IF(NPD.EQ.NSKS) GO TO 4343
      KPP=NPD+1
      WRITE(NPRNT,227) (NSINK(K),K=KPP,NSKS)
 4343 IF(NCTS) 135,135,125
  125 DO 130 J=1,NCTS
  130 KOUNT(J)=0
  135 IM=5
      NOQ=NN
      MXC=32
      IMM=2
      ATRIB(2)=0.0
      MSTOP=0
      DO 9 I = 1,NPRMS
      READ (NCRDR,106) (PARAM(I,J),J=1,4)
  106 FORMAT(4F10.4)
    9 CONTINUE
      WRITE(NPRNT,21)
      WRITE(NPRNT,26)
      DO 50 I = 1,NPRMS
   50 WRITE(NPRNT,28) I,(PARAM(I,J),J=1,4)
      ISEED=JSEED
      RNUM=DRAND(ISEED)
      TNOW=0.0
C
C*****INITIALIZE NSET
C
      KOL=7777
      KOF = 8888
      KLE = 9999
      MX  = IM +1
      MXX=IM+2
      MAXQS = ID * IMM
      MAXNS = ID * MXX
      DO 35 J=1,MAXCS
   35 QSET(J)=0.0
      DO 40 J=1,MAXNS
   40 NSET(J)=0
      DO 45 I=1,ID
      INDX = I * MXX
      NSET(INDX - 1) = I + 1
   45 NSET(INDX)=I-1
      NSET(MAXNS - 1) = KOF
      DO 3 K = 1,NOQ
      NQ(K)=0
      MFE(K)=0
    3 MLE(K)=0
```

```
      MFA = 1
      NDTL=NSKD+1
      NDTU=NDTL+NDT-1
      IF(NUMGQ .EQ. 0) GO TO 6421
      WRITE(NPRNT,9025)
      DO 2 I=1,NUMGQ
      READ(60,4) IGATE(I,1), IGATE(I,2), ITEST(I)
      WRITE(NPRNT,9011) IGATE(I,1), IGATE(I,2), ITEST(I)
    2 CONTINUE
 9025 FORMAT(//42X,16H**GATED QUEUES**,//16X,5HQUEUE,12X,7HSPECIAL,12X,
     1 6HNUMBER/16X,4HNODE,13X,4HNODE,15X,7HLEAVING)
 9011 FORMAT(19X,I3,15X,I3,15X,I3)
    4 FORMAT(3I3)
 6421 DO 300 JS = 1,ID
C
C*****DATA CARD TYPE  FIVE
C
      READ(NCRDR,1110) ATRIB(1),JQ,(JTRIB(JK),JK=1,IM)
 1110 FORMAT(F8.3,6I3)
      IF(JQ) 310,15,320
  310 NRUNS=-1
      GO TO 300
  320 J=JTRIB(2)
      JD=JTRIB(3)
      GO TO (66,66,66,66,65,66,67,68,69),JD
   65 PARAM(J,4) = ALOG((PARAM(J,4)/PARAM(J,1))**2 + 1.0)
      PARAM(J,1) = ALOG(PARAM(J,1)) - .5 * PARAM(J,4)
      PARAM(J,4) = SQRT(PARAM(J,4))
      IF(PARAM(J,2)) 7020,7020,7021
 7020 PARAM(J,2)=-1.0E20
      GO TO 7023
 7021 PARAM(J,2)=ALOG(PARAM(J,2))
 7023 PARAM(J,3)=ALOG(PARAM(J,3))
      GO TO 66
   67 CALL BETAXF
      GO TO 66
   68 TEMP = PARAM(J,4) ** 2
      PARAM(J,4) = PARAM (J,1) **2 / TEMP
      PARAM(J,1) = PARAM(J,1) / TEMP
      GO TO 66
   69 CALL PERTXF
   66 IPOINT = 0
      DO 9023 M = 1,NUMGQ
      IF(JTRIB(1) .EQ. IGATE(M,2)) IPOINT = M
 9023 CONTINUE
      IF(IPOINT .GT.0) JTRIB(4) = -IGATE(IPOINT,1)
      IF(NDT) 642,642,639
C*****NDTU--LOCATION FOR ARRAY STORAGE FOR DELAY NODES (UPPER)
  639 DO 640 I=NDTL,NDTU
      IF(JTRIB(1)-NSINK(I)) 640,641,640
C*****HERE WE TEST IF END NODE OF ACTIVITY IS A DELAY NODE
  640 CONTINUE
      GO TO 642
  641 JTRIB(4)=-IMN-1
```

```
      GO TO 1320
  642 IF(NQN) 1320,1320,643
  643 DO 647 I=NQL,NQU
      IF(JTRIB(1)-NSINK(I)) 644,645,644
C*****TEST IF ACTIVITY IS COMING INTO A QUEUE NODE;IF SO, SET ACCNT
C***** NUMBER = -1
  644 IF(JQ-NSINK(I)) 647,646,647
C*****TEST IF ACTIVITY IS LEAVING A QUEUE NODE; IF YES, IT IS A
C***** PROCESSOR; SET ACCNT NUMBER = -(Q-NODE NUMBER)
  647 CONTINUE
      GO TO 1320
  645 JTRIB(4)=-1
      GO TO 1320
  646 JTRIB(4)=-JQ
C $$$ IF THE Q IS A GATED Q, PUT -(Q-NODE NUMBER) WITH SPECIAL NODE
C $$$ INSTEAD OF WITH THE NODE THAT IS ON END OF BRANCH WITH QUEUE
      DO 9024 M = 1,NUMGQ
      IF(JQ .EQ. IGATE(M,1)) JTRIB(4) = 0
 9024 CONTINUE
 1320 CALL FILEM(JQ,NSET,QSET)
  300 CONTINUE
   15 NCLCT=NSKS*(NCTS+1) + NQN * 3
      NHIST=NSKS + NQN
      DO 18 I = 1,NCLCT
      DO 17 J =1,3
   17 SUMA(I,J) = 0.
      SUMA(I,4) = 1.0E20
   18 SUMA(I,5)= -1.0E20
      IF(NHIST)110,9022,118
  110 NRUNS=-2
      GO TO 9022
  118 DO 380 K = 1,NHIST
      DO 380 L = 1,MXC
  380 JCELS(K,L) = 0
 9022 WRITE(NFRNT,24)
      DO 80 JQ=2,NOQ
      PROB = 0.0
      LINE = MFE(JQ)
      IF(LINE)80,80,82
   82 IB = (LINE-1) * MXX + 1
      NT = NTYPE(JQ)
      IE = IB + IM-1
      IQ = (LINE-1) * IMM + 1
      DPROB = QSET(IQ)
      GO TO (72,73,72,73),NT
   73 QSET(IQ)=QSET(IQ)+PROB
      PROB = PROB + DPROB
   72 WRITE(NPRNT,12) JQ,(NSET(I),I=IB,IE),DPROB
      LINE = NSET(IE+1)
      IF(LINE - 7777)82,480,84
   84 WRITE(NPRNT,199)
  480 GO TO (80,481,80,481),NT
  481 IF (PROB-.998)482,8077,8077
 8077 IF (PROB-1.002)80,80,482
 482 WRITE(NPRNT,483) JQ
 483 FORMAT(/5X,4HNODE,I4,59H HAS OUTPUT BRANCHES WHOSE PROBABILITIES D
    10 NOT SUM TO ONE./)
  80 CONTINUE
  21 FORMAT(1H1)
  22 FORMAT(32X,23HGERT SIMULATION PROJECT,I4,4H BY ,6A2/45X,4HDATE,
    1I3,1H/,I3,1H/,
    2I5//42X,23H**NETWORK DESCRIPTION**//44X,20HNODE CHARACTERISTICS/)
  24 FORMAT(//42X,24H**ACTIVITY DESCRIPTION**//16X5HSTART,6X3HEND,6X,
    19HPARAMETER,6X,12HDISTRIBUTION,6X,5HCOUNT,6X,8HACTIVITY,6X,
    2 11HPROBABILITY/
    317X4HNODE,6X,4HNODE,7X,6HNUMBER,11X4HTYPE,11X4HTYPE,7X,6HNUMBER/)
  12 FORMAT(I19,I11,I12,I16,2I14,F17.4)
 199 FORMAT(///36X26HERROR EXIT, TYPE 70 ERROR.)
  26 FORMAT(//39X23H**ACTIVITY PARAMETERS**//16X9HPARAMETER32X,10HPARAM
    1ETERS/17X6HNUMBER,16X1H1,13X1H2,13X,1H3,13X1H4/)
  28 FORMAT(I21,7X,4F14.4)
     DO 1024 J=1,IMN
1024 NPO(J)=0
     IF(NNM) 1020,1020,1023
1023 WRITE(NPRNT,1004)
1004 FORMAT(//39X25H**NETWORK MODIFICATIONS**//12X,8HACTIVITY,10(1X,9HN
    1ODE FILE)/)
     NPOS=0
1026 NPOS=NPOS+1
     READ(NCRDR,1005) NACTN,NABYA
1005 FORMAT(25I3)
     IF(NACTN) 1036,1036,1037
1037 NPO(NACTN)=NPOS
     DO 1038 I=1,22,2
     IF(NABYA(I)) 1040,1040,1039
1039 NABA(NPOS)=NABYA(I)
     NABA(NPOS+1)=NABYA(I+1)
1038 NPOS=NPOS+2
1040 NABA(NPOS)=0
     IM1=I-1
     WRITE(NPRNT,1006) NACTN,(NABYA(J),J=1,IM1)
1006 FORMAT(I17,2X,10(I5,I5))
     IF(NPOS-IMN)1026,1030,1030
1030 NRUNS=-3
     GO TO 1026
1036 NABA(NPOS)=0
1020 IF(NPROJ) 1591,1592,1592
1591 IMN1=IMN-1
     READ(NCRDR,10) NABA(IMN),NABA(IMN1)
1592 WRITE(NPRNT,21)
 221 FORMAT(     15X22HHIGHEST NODE NUMBER IS,I5,/15X,25HNUMBER OF SOURC
    1E NODES IS,I5/15X,23HNUMBER OF SINK NODES IS,I5/15X41HNUMBER OF NO
    2DES TO REALIZE THE NETWORK IS,I5/15X,23HSTATISTICS COLLECTED ON,I5
    3,6H NODES/15X,27HNUMBER OF PARAMETER SETS IS,I5/15X24HINITIAL RAND
    4OM NUMBER IS, I8,F10.4/)
 223 FORMAT(/15X,19HSOURCE NODE NUMBERS/17X,10(10I5))
 225 FORMAT(/15X,17HSINK NODE NUMBERS/17X,10(10I5))
 227 FORMAT(/15X,34HSTATISTICS COLLECTED ALSO ON NODES/17X,10(10I5/))
```

```
  229 FORMAT(/5H NODE,4X,6HNUMBER,5X,18HNUMBER OF RELEASES,3X,6HOUTPUT,4
     1X,15HREMOVAL DESIRED,2X,16HSTATISTICS BASED/8X 8HRELEASES,8X,10HFO
     2R REPEAT,7X,4HTYPE,6X,14HAT REALIZATION,3X,15HON REALIZATIONS/)
  233 FORMAT(I5,I9, I18,13X,A1,13X,A1,14X,A3,A4,I4)
      IF(NRUNS.LT.0.AND.NRUNS.GT.-4)CALL ERROR(NRUNS,NSET,QSET)
      END
```

```
       SUBROUTINE BETAXF
C
C*****THIS ROUTINE CONVERTS THE GIVEN BETA PARAMETERS, MEAN + VARIANCE
C*****TO (1) THE MEAN AND VARIANCE OF A BETA DISTRIBUTION OVER THE
C*****INTERVAL (0,1), (2) THE K1 + K2 VALUES NEEDED TO OBTAIN A BETA
C*****VARIATE FROM THE RATIO  OF TWO GAMMA VARIATES
C
C      PARAM(J,1) CONTAINS THE MEAN+ AT CLOSE IT WILL CONTAIN K1
C      PARAM(J,2) CONTAINS THE LOWER LIMIT
C      PARAM(J,3) CONTAINS THE UPPER LIMIT
C      PARAM(J,4) CONTAINS THE VARIANCE# AT CLOSE IT WILL CONTAIN K2
C
       COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
      1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
      2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
       COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
       J=JTRIB(2)
       BMEAN = (PARAM(J,1) - PARAM(J,2))/(PARAM(J,3) - PARAM(J,2))
       BVAR=(PARAM(J,4)/(PARAM(J,3)-PARAM(J,2)))**2
       PARAM(J,1) = BMEAN * (BMEAN * (1.0 - BMEAN)/BVAR - 1.0)
       PARAM(J,4) = PARAM(J,1) * (( 1.0 - BMEAN)/BMEAN)
       RETURN
       END
```

```
       SUBROUTINE PERTXF
C
       COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
      1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
      2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
       COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
       J=JTRIB(2)
       PARAM(J,1) = (4.*PARAM(J,1)+ PARAM(J,2)  + PARAM(J,3))/6.0
       PARAM(J,4)=(PARAM(J,3)-PARAM(J,2))/6.0
       CALL BETAXF
       RETURN
       END
```

4.3 Overlay 2

```
      PROGRAM GASP
      DIMENSION NRELS(50), NRELS2(100)
      INTEGER PLOT(50)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,MCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMON NDT,NDTL,NDTU,NSKD,NSTND,XSTUS(99)
      COMMON NQN,NQL,NQU,NSTUS(100),SBUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEQ(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
      COMMON IGATE(50,2),NUMGQ,ITEST(50)
      COMMON NSET(2275),QSET(1350)
C
C
C*****INITIALIZE FOR THIS RUN OF THE SIMULATION
C
      TOTIM=0.0
 6911 DO 9994 I=1,100
 9994 NRELS2(I)=606060608
      IF(SEED.NE.0.0.AND.NPROJ.LT.0)WRITE(NPRNT,9996)NRUN,(IMNQ(I),I=1,K
     11)
 9996 FORMAT(8H RUN NO.  I4,5HTIME  50I2)
C $$$ IF Q IS GATED, INITIALIZE NSTUS TO CONTAIN HOW MANY TO LET OUT
C $$$ BEFORE GATING. NSTUS IS SET EQUAL TO -TIEST +1 IN ORDER TO USE
C $$$ THE EXISTING PROGRAM.  WHEN NSTUS REACHES +1 IT GATES.
      DO 9004 I = NQL,NQU
      K = I-NSKS
 9004 NSTUS(K) = 0
      DO 9993 I=1,50
 9993 NRELS(I)=0
      TNOW1=0.0
      INT=1
      DO 6980 MU=1,IM
 6980 JTRIB(MU)=0
      DO 6775 M=1,IMM
 6775 ATRIB(M)=0.
      CALL FILEM(1,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      TNOW=0.
      NSKSR=NSKST
      DO 85 J=1,NCTS
   85 KOUNT(J)=0
      DO 37 IJ=2,NN
      NFTBU(IJ)=IJ
      NRELP(IJ)=IABS(NRELP(IJ))
   37 NREL(IJ)=NRELP(IJ)
      IF(NQN) 11,11,843
C
C*****REINITIALE QUEUE NODE POINTERS
C
```

89

```
  843 DO 2860 K=1,NGN
      MLEQ(K)=0
 2860 MFEQ(K)=0
      DO 2861 K=1,IMNN
      QATT(K)=0.0
 2861 NPTR(K)=K+1
      NPTR(IMNN)=8888
      MFAQ=1
      DO 6971  K=NQL,NQU
      NEND = NSINK(K)
      IPOINT = 0
      DO 9000 J = 1,NUMGQ
      IF(NEND .EQ. IGATE(J,1)) IPOINT = J
 9000 CONTINUE
      IF(IPOINT .EQ. 0) GO  TO 9002
      KK = K-NSKS
      NSTUS(KK) = -ITEST(IPOINT) +1
 9002 IF(NRELP(NEND)) 6971,6971,6972
 6972 NIQ=NRELP(NEND)
      ATRIB(2)=0.0
      CALL SCHAT(NEND,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      KK=K-NSKS
      NSTUS(KK) = NSTUS(KK) +1
      DO 6973 IQ=1,NIQ
      IF (MFAQ-8888) 6859,23,23
   23 CALL ERROR(23,NSET,QSET)
      GO TO 9990
 6859 MFAT=NPTR(MFAQ)
      QATT(MFAQ)=ATRIB(2)
      IF(MFEQ(KK)) 6860,6860,6863
 6860 MFEQ(KK)=MFAQ
      GO TO 6864
 6863 MKKK=MLEQ(KK)
      NPTR(MKKK)=MFAQ
 6864 MLEQ(KK)=MFAQ
      NPTR(MFAQ)=7777
 6973 MFAQ=MFAT
 6971 CONTINUE
C
C*****OBTAIN NEXT EVENT WHICH IS FIRST ENTRY IN FILE 1.  ATRIB(1) IS
C*****EVENT TIME, JTRIB(1) IS EVENT CODE
C
   11 CALL RMOVE(MFE(1),1,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      TNOW=ATRIB(1)
      NEND=JTRIB(1)
C
C*****IF END NODE IS ZERO START NETWORK BY SCHEDULING ACTIVITIES
C*****FROM EACH SOURCE NODE
C
      IF(NEND)4,4,5
    4 DO 10 N=1,NSRC
      M=NSORC(N)
      NREL(M)=NREL2(M)
      CALL SCHAT(M,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
   10 CONTINUE
      GO TO 42
C
C*****REDUCE NO. OF RELEASES FOR END NODE BY 1.
C*****TEST TO SEE IF NODE IS RELEASED.
C
    5 NREL(NEND)=NREL(NEND)-1
   20 IF(NPROJ)511,512,512
C
C*****CHECK FOR TRACE ON THIS RUN
C
  511 IF(NRUN-NABA(IMN)) 512,514,514
  514 IMN1=IMN-1
      IF(NRUN-NABA(IMN1)) 515,515,512
  598 FORMAT (1X7HAT TIME,F10.2,17H ACTIVITY ON NODE,I5,16H WITH ATTRIBU
     1TES,4I5,20H WAS REALIZED ON RUN,I5)
C
C****** TESTS TO SEE IF TSD IS RUN AND IF TIME HAS CHANGED SINCE
C****** LAST PRINTING.
C
  515 IF(SEED)9998,9997,9998
 9998 IF(TNOW.EQ.TNOW1) GO TO 3
C
C***** FROM STATEMENT NO 6 TO 3, THIS IS THE SECTION WHICH SETS-UP
C***** THE PRINTING IF TSD. IF NRELS(NO. IN NODE) IS ZERO, A PERIOD IS
C***** PRINTED, IF 1, THEN T IF 2-99 THEN THE ACTUAL NO., IF OVER 99,
C***** THEN A LETTER STARTING FROM A IS PRINTED GOING BY HUNDREDS.
C
      DO 12 I1=1,K1
      IF(NRELS(I1)-1)13,14,15
   13 PLOT(I1)=60336060B
      GO TO 12
   14 PLOT(I1)=60636060B
      GO TO 12
   15 IF(NRELS(I1)-10)16,19,19
   16 PLOT(I1)=NRELS(I1)*10000B + 60006060B
      GO TO 12
   19 IF(NRELS(I1)-99)17,17,18
   17 NRELS1=NRELS(I1)/10
      N1=NRELS(I1)-NRELS1*10
      PLOT(I1)=NRELS1*1000000B+N1*10000B+00006060B
      GO TO 12
   18 N1=20B+NRELS(I1)/100
      IF(N1.GT.31B)N1=31B
      PLOT(I1)=N1*1000000B
   12 CONTINUE
      WRITE(NPRNT,21)TNOW1,(PLOT(I),I=1,K1)
   21 FORMAT(6X,F9.3,2X,50A2)
      WRITE(NPRNT,24)(NRELS2(I),I=1,100)
   24 FORMAT(17X,100A1)
      DO 9995 I=1,100
 9995 NRELS2(I)=60606060B
      TNOW1 = TNOW
      GO TO (3,102), INT
C
C***** FROM STATEMENT NO. 3 TO 33, NRELS IS BEING SET UP. IT TESTS
C***** TO SEE IF THE NOCE IS A QUEUE NODE(DO LOOP 34), AND SETS UP NRELS
C****** ACCORDINGLY. 56B IS AN DOWNWARD ARROW, 55 IS UPWARD.
C
    3 I1=1
      DO 33 I=2,NOQ
      IF(IMNQ(I1).NE.I) GO TO 33
      DO 34 J=NQL,NQU
      IF(NSINK(J).EQ.I) GO TO 35
   34 CONTINUE
      NR=NRELS(I1)
      NRELS(I1)=NRELP(I)-NREL(I)
      IF(NRELP(I).LT.0) NRELS(I1)=NREL2(I)-NREL(I)
   36 IF(NRELS(I1)-NR) 31,30,38
   35 NR=NRELS(I1)
      NRELS(I1)=NREL(I)
      IF(I.EQ.NEND) NRELS(I1)=NRELS(I1)+1
      GO TO 36
   31 I2=2*I1
      NRELS2(I2)=56606060B
      GO TO 30
   38 I2=2*I1-1
      NRELS2(I2)=55606060B
   30 I1=I1+1
   33 CONTINUE
      GO TO 512
 9997 WRITE(NPRNT,598) TNOW, JTRIB, NRUN
  512 GO TO (513,102),INT
  513 IF(JTRIB(5))665,665,66
C
C*****NETWORK MODIFICATION SECTION
C
   66 JJJ=JTRIB(5)
      IF(NPO(JJJ)) 665,665,67
   67 NPOS=NPO(JJJ)
   69 NDN=NABA(NPOS)
      NTBI=NABA(NPOS+1)
      DO 669 N=1,NOQ
      IF(NFTBU(N).EQ.NDN) NFTBU(N)=NTBI
  669 CONTINUE
      IF(NABA(NPOS+2)) 665,665,68
   68 NPOS=NPOS +2
      GO TO 69
  665 IF(JTRIB(4)) 701, 65,60
C
C*****COUNTER TYPE SECTION
C
   60 L=JTRIB(4)
      KOUNT(L)=KOUNT(L)+1
      GO TO 65
  701 IF(JTRIB(4)+IMN) 702,705,705
C
C*****DELAY NODE SECTION
C
  702 DO 704 K=NDTL,NDTU
      IF(NEND-NSINK(K)) 704,703,704
  704 CONTINUE
      WRITE(NPRNT,751) NEND,TNOW,NRUN
  751 FORMAT(1X20HERROR FOR DELAY NODE,I5,8H AT TIME,F10.4,7H ON RUN,I5)
      CALL ERROR(24,NSET,QSET)
      GO TO 9990
  703 IF(XSTUS(K)) 749, 65, 65
  749 XSTUS(K)=TNOW
      GO TO  65
C
C*****QUEUE NODE SECTION,  NNOQ = QUEUE NODE NUMBER
C
  705 IF(JTRIB(4)+1) 707,708,706
  706 NNOQ=NEND
      NREL(NEND)=NREL(NEND)+1
C*****NNOQ--NODE NUMBER OF THE THE QUEUE (ASSOCIATED WITH PROCESSOR OR
C***** ACTIVITY COMING INTO A QUEUE
      GO TO 708
  707 NNOQ=-JTRIB(4)
  708 DO 709 K=NQL,NQU
      IF(NNOQ-NSINK(K)) 709,710,709
  709 CONTINUE
      WRITE(NPRNT,752) NNOQ,TNOW,NRUN
  752 FORMAT(1X20HERROR FOR QUEUE NODE,I5,8H AT TIME,F30.4,7H ON RUN,I5)
      CALL ERROR(21,NSET,QSET)
      GO TO 9990
  710 KK=K-NSKS
      XX=NREL(NNOQ)
      SQUE(KK)=SQUE(KK)+XX*(TNOW-TLC(KK))
      IF(NSTUS(KK)) 711,711,712
  712 SBUS(KK)=SBUS(KK)+TNOW-TLC(KK)
  711 TLC(KK)=TNOW
C
C*****TEST FOR A PROCESSOR OR ARRIVAL INTO A QUEUE
C
      IF(JTRIB(4)+1)713,714,714
C
C*****QUEUE PROCESSOR SECTTON
C
  713 IF(NREL(NNOQ)) 715,715,716
 715  NSTUS(KK) = NSTUS(KK) -1
C****THE PROCESSOR IS EMPTY
      GO TO 65
C
C*****UPDATE QUEUE FILE
C
  716 NREL(NNOQ)=NREL(NNOQ)-1
      MFEQT=MFEQ(KK)
      ATRBT=ATRIB(2)
      ATRIB(2)=QATT(MFEQT)
      QATT(MFEQT)=0.0
      IF(NPTR(MFEQT).LT.7777) GO TO 870
      MFEQ(KK)=0
      MLEQ(KK)=0
      GO TO 872
  870 MFEQ(KK)=NPTR(MFEQT)
  872 NPTR(MFEQT)=MFAQ
      MFAQ=MFEQT
      CALL SCHAT(NNOQ,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      ATRIB(2)=ATRBT
      GO TO 65
C
C*****QUEUE ARRIVAL SECTION
C
  714 IF(NREL(NNOQ)) 717,717,718
  717 IF(NSTUS(KK)) 719,719,718
 719  NSTUS(KK) = NSTUS(KK) +1
C
C*****IF SERVER IS IDLE, SCHEDULE A SERVICE COMPLETION TIME
C
      CALL SCHAT(NNOQ,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      GO TO 42
C
C*****CHECK FOR BALKERS
C
  718 IF(NREL(NNOQ)-NREL2(NNOQ)) 720,1665,1665
  720 NREL(NNOQ)=NREL(NNOQ)+1
      IF (MFAQ-8888) 859,27,27
   27 CALL ERROR(27,NSET,QSET)
      GO TO 9990
  859 MFAT=NPTR(MFAQ)
      QATT(MFAQ)=ATRIB(2)
      IF(MFEQ(KK)) 860,860,863
  860 MFEQ(KK)=MFAQ
      GO TO 864
  863 MKKK=MLEQ(KK)
      NPTR(MKKK)=MFAQ
  864 MLEQ(KK)=MFAQ
      NPTR(MFAQ)=7777
  865 MFAQ=MFAT
      GO TO 42
C
C*****INCREMENT NUMBER OF BALKERS
C
 1665 XBALK(KK)=XBALK(KK)+1
      IF(JSINK (K))42,42,1493
 1493 ATRIB(1)=TNOW
      JTRIB(1)=JSINK(K)
      DO 1494 MAN=2,5
 1494 JTRIB(MAN)=0
C
```

```
C*****FILE BALKER ACTIVITY IN EVENT FILE
C
      CALL FILEM(1,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      GO TO 42
C
C*****IF THE NUMBER OF RELEASES IS NEGATIVE, NODE IS A MARK NODE
C
   65 IF(NREL(NEND))9,7,42
    9 ATRIB(2)=TNOW
      GO TO 52
C
C*****TEST TO SEE IF STATISTICS ARE TO BE COLLECTED ON THE NODE.
C
    7 NRELP(NEND)=-NRELP(NEND)
      DO 8 K=1,NSKS
      IF(NEND-NSINK(K))8,39,8
   39 IF(JSINK(K)) 40,50,438
  438 IF(JSINK(K)-1) 441,441,439
    8 CONTINUE
   50 NT=NTYPE(NEND)
      NREL(NEND)=NREL2(NEND)
      GO TO (52,52,55,55),NT
   55 ATRBB=ATRIB(2)
   54 KCOL=NFIND(NEND,1,1,NSET,QSET)
      IF (NRUNS.LT.0) GO TO 9990
      IF (KCOL)53,53,56
   56 CALL RMOVE(KCOL,1,NSET,QSET)
      IF (NRUNS.LT.0) GO TO 9990
      GO TO 54
   53 ATRIB(2)=ATRBB
   52 NODE=NFTBU(NEND)
      IF (NQ(NODE)) 25,42,51
   25 CALL ERROR(25,NSET,QSET)
      GO TO 9990
   51 CALL SCHAT(NODE,NSET,QSET)
      IF (NRUNS.LT.0) GO TO 9990
   42 GO TO 11
  439 JSINK(K)=-JSINK(K)
      NSNR(K)=NSNR(K)+1
      GO TO 40
C
C*****REDUCE NO. OF SINK NODES REALIZED BY ONE.  COLLECT STATISTICS.
C
  441 JSINK(K)=0
   40 IF(K-NPD) 440,440,742
  440 NSKSR=NSKSR-1
  742 IF(JTRIB(4)+IMN) 743,442,442
C
C*****DELAY NODE STATISTICS
C
  743 YY = TNOW - XSTUS(K)
      XSTUS(K) = -1.0
      GO TO 1442
```

```
C
C*****FIRST OR ALL STATISTICS
C
  442 YY=TNOW
      IF(IABS(JSINK(K))-3) 1442,1443,1444
C
C*****BETWEEN STATISTICS
C
 1443 YY=TNOW-XSTUS(K)
      XSTUS(K)=TNOW
      IF(XSTUS(K)) 45,1442,1442
C
C*****INCREMENT STATISTICS
C
 1444 YY=TNOW-ATRIB(2)
 1442 CALL HISTO(YY,K)
      IF(NRUNS.LT.0)GO TO 9990
      NL=K+(K-1)*NCTS
      CALL COLCT(YY,NL,NSET,QSET)
      IF (NRUNS.LT.0) GO TO 9990
      IF(NCTS) 45,45,46
C
C*****COUNTER STATISTICS
C
   46 LINE=MFE(1)
      IF(LINE) 346,346,308
  308 INDX=(LINE-1)*IMM+1
      IF(TNOW-QSET(INDX)) 346,310,346
  310 INDX=(LINE-1)*MXX+4
      IF(NSET(INDX)) 312,312,313
  313 L=NSET(INDX)
      KOUNT(L)=KOUNT(L)+1
      NSET(INDX)=0
  312 LINE=LINE*MXX+1
      LINE=NSET(LINE)
      IF(LINE-7777) 308,346,346
  346 DO 70 I=1,NCTS
      NL=NL+1
      XC=KOUNT(I)
   70 CALL COLCT(XC,NL,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
C
C*****TEST TO SEE IF NETWORK SIMULATION IS COMPLETE.
C
   45 IF(NSKSR) 22,100,50
   22 CALL ERROR(22,NSET,QSET)
      GO TO 9990
C
C*****NETWORK HAS BEEN SIMULATED ONE MORE TIME.  ADD TNOW TO TOTAL TIME.
C
  100 TOTIM=TOTIM+TNOW
C
C***** SETS UP FOR THE FINAL PRINTING OF TSD.
C
```

```
      IF(SEED.EQ.0.0.OR.NPROJ.GE.0) GO TO 101
      TNOW=TNOW+1.0
      INT=2
      NRELS(NEND)=0
      GO TO 20
  102 TNOW=TNOW-1.0
  101 IF(NQN) 797,797,798
C
C*****COLLECT STATISTICS ON ALL Q-NODES AND PROCESSORS
  798 KLL=NSKS*(NCTS+1)+1
      KLH=NSKS+1
      DO 730 I=1,NQN
      NSTUS(I)=0
      SQUE(I)=SQUE(I)/TNOW
      CALL COLCT(SQUE(I),KLL,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      CALL HISTO(SQUE(I),KLH)
      IF(NRUNS.LT.0)GO TO 9990
      SQUE(I)=0.0
      TLC(I)=0.0
      SBUS(I)=SBUS(I)/TNOW
      CALL COLCT (SBUS(I),KLL+1,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      SBUS(I)=0.0
      XBALK(I)=XBALK(I)/TNOW
      CALL COLCT(XBALK(I),KLL+2,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      XBALK(I)=0.0
      KLL=KLL+3
  730 KLH=KLH+1
  797 DO 80 K=1,NSKS
      IF(JSINK(K)) 81,82,80
   81 JSINK(K)=-JSINK(K)
      GO TO 80
   82 JSINK(K)=1
   80 CONTINUE
      DO 1998 M=1,NSKS
 1998 XSTUS(M)=-1.0
C
C*****TEST TO SEE IF ALL RUNS HAVE BEEN MADE.
C
      IF(NRUNS-NRUN)110,110,115
C
C*****REINITIALE FOR ANOTHER RUN BY REMOVING ALL EVENTS FROM EVENT
C*****FILE AND RESETTING NETWORK VALUES.
C
  115 IF (NQ(1)) 26,120,116
   26 CALL ERROR(26,NSET,QSET)
      GO TO 9990
  116 CALL RMOVE(MFE(1),1,NSET,QSET)
      IF(NRUNS.LT.0)GO TO 9990
      GO TO 115
  120 NRUN=NRUN+1
      GO TO 6911
  110 NRUNS=1
      TNOW=TOTIM
C
C*****IS IT TIME TO STOP
C
      CALL SUMRY(NSET,QSET)
 9990 END
```

```
      FUNCTION BETA(J)
C
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
C
      IF(PARAM(J,1))10,10,20
   10 WRITE(NPRNT,11) J
   11 FORMAT (/50H STANDARD FIX TAKEN:   K  WAS SET = 1.0 FOR BETA   ,I4)
      PARAM(J,1)=1.0
   20 IF(PARAM(J,4))110,110,30
  110 WRITE(NPRNT,11) J
      PARAM(J,4)=1.0
   30 P = PGAMMA(PARAM(J,1),NPRNT,J,ISEED)
      BETA = P/(P + PGAMMA(PARAM(J,4),NPRNT,J,ISEED))
      BETA = BETA * (PARAM(J,3) - PARAM(J,2)) + PARAM(J,2)
      RETURN
      END
```

```
      SUBROUTINE COLCT (X,N,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      IF (N) 2,2,1
    2 CALL ERROR(90,NSET,QSET)
      RETURN
    1 IF (N- NCLCT) 3,3,2
    3 SUMA(N,1) = SUMA(N,1)+X
      SUMA(N,2) = SUMA(N,2)+X*X
      SUMA(N,3) = SUMA(N,3)+1.0
      IF(X.LT.SUMA(N,4))SUMA(N,4)=X
      IF(X.GT.SUMA(N,5))SUMA(N,5)=X
      RETURN
      END
      FUNCTION ERLNG (J)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      K = PARAM(J,4) + .001
      IF(K-1) 8,10,10
    8 WRITE(NPRNT,20) J
   20 FORMAT (/26HSTANDARD FIX TAKEN: PARAM(,I5,17H4) WAS SET TO 1.0)
      K=1
   10 RNUM=1.
      DO 2 I = 1,K
    2 RNUM=DRAND(ISEED)*RNUM
      ERLNG =         - PARAM(J,1)*ALOG(RNUM)
      IF(ERLNG-PARAM(J,2))7,5,6
    7 ERLNG  = PARAM (J,2)
    5 RETURN
    6 IF(ERLNG  - PARAM (J,3))5,5,4
    4 ERLNG  = PARAM (J,3)
      RETURN
      END
```

```
      FUNCTION NFIND(NVAL,JQ,JATT,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      NEXTK=MFE(JQ)
      IF(NEXTK) 16,1,20
   16 CALL ERROR(89,NSET,QSET)
    1 NFIND=0
      RETURN
   20 INDX=(NEXTK-1)*MXX +JATT
      IF(NSET(INDX)-NVAL) 4,21,4
    4 INDS=NEXTK*MXX -1
      NEXTK=NSET(INDS)
      IF(NEXTK-7777) 20,1,1
   21 NFIND=NEXTK
      RETURN
      END
      SUBROUTINE HISTO(X1,N)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      IF (N-NHIST) 11,11,2
    2 WRITE(NPRNT,250) N
  250 FORMAT(19H ERROR IN HISTOGRAM,I4//)
      CALL EXIT
   11 IF(N)2,2,3
C
C*****TRANSLATE X1 BY SUBTRACTING A IF X.LE.A THEN ADD 1 TO FIRST CELL
C
    3 X=X1-XLOW(N)
      IF (X)6,7,7
    6 IC = 1
      GO TO 8
C
C*****DETERMINE CELL NUMBER IC. ADD 1 FOR LOWER LIMIT CELL AND 1 FOR
C*****TRUNCATION
C
    7 IC=X/ABS(WIDTH(N))+2.+.0001
      IF(IC - MXC     )8,8,9
    9 IC = MXC
    8 JCELS(N,IC) = JCELS(N,IC) + 1
      RETURN
      END
      FUNCTION GAMMA5( ALPHA,GMIN,GMAX,TK,NPRNT,JP,ISEED)
C
C*****THIS ROUTINE CALCULATES A GAMMA VARIATE.  REFERENCE [COMPUTER
C*****SIMULATION TECHNIQUES[ NAYLOR, ET AL.,WILEY AND SONS,PP 87-89
C
      IF (ALPHA) 40,40,50
   40 WRITE(NPRNT,41) JP
   41 FORMAT (/49HSTANDARD FIX TAKEN: ALPHA WAS SET = 1.0 FOR GAMMA,I4)
      ALPHA=1.0
   50 IF (TK) 60,60,70
   60 WRITE (NPRNT, 61) JP
   61 FORMAT (/49HSTANDARD FIX TAKEN:   K   WAS SET = 1.0 FOR GAMMA,I4)
      TK=1.0
   70 GAMMA = 1.0
      K1 = TK
      R = DRAND(ISEED)
      TK1 = K1
      IF (R - (TK- TK1)) 90,90,100
   90 K1 = K1 + 1
  100 DO 110 I=1, K1
      R = DRAND(ISEED)
  110 GAMMA = GAMMA * R
      GAMMA5 = -ALOG(GAMMA)/ALPHA
      IF (GAMMA5 - GMIN) 120,130,140
  120 GAMMA5 = GMIN
  130 RETURN
  140 IF(GAMMA5 - GMAX) 130,130,150
  150 GAMMA5 = GMAX
      RETURN
      END
```

```
      FUNCTION RLOGN (J)
C
C*****THE PARAMETERS USED WITH RLOGN ARE THE MEAN AND STANDARD DEVIATION
C*****OF A NORMAL DISTRIBUTION
C
      VA= RNORM (J)
      RLOGN=EXP(VA)
      RETURN
      END
      SUBROUTINE NPCSN(J,NPSSN)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      NPSSN = 0
      P = PARAM (J,1)
      IF (P-6.0) 2,2,4
    2 Y = EXP (-P)
      X = 1.0
    3 RNUM=DRAND(ISEED)
      X=X*RNUM
      IF (X-Y) 6,8,8
    8 NPSSN = NPSSN+1
      GO TO 3
    4 TEMP=PARAM (J,4)
      PARAM(J,4) = (PARAM(J,1))**.5
      NPSSN=RNORM(J) +.5
      PARAM (J,4)=TEMP
      IF(NPSSN)4,6,6
    6 KK=PARAM (J,2)
      KKK=PARAM (J,3)
      NPSSN=KK+NPSSN
      IF(NPSSN-KKK)7,7,9
    9 NPSSN = PARAM (J,3)
    7 RETURN
      END
```

```
      FUNCTION PGAMMA(TK,NPRNT,JP,ISEED)
C
C*****THIS ROUTINE CALCULATES A GAMMA VARIATE.  REFERENCE [COMPUTER
C*****SIMULATION TECHNIQUES[ NAYLOR, ET AL.,WILEY AND SONS,PP 87-89
C
      IF (TK) 60,60,70
   60 WRITE(NPRNT,61) JP
   61 FORMAT (/49HSTANDARD FIX TAKEN!   K  WAS SET = 1.0 FOR PGAMMA,I4)
      TK=1.0
   70 GAMMA = 1.0
      K1 = TK
      TK1 = K1
      R = DRAND(ISEED)
      IF (R -(TK - TK1)) 90,90,100
   90 K1 = K1 + 1
  100 DO 110 I = 1,K1
      R = DRAND(ISEED)
  110 GAMMA = GAMMA * R
      PGAMMA = -ALOG(GAMMA)
      RETURN
      END
      SUBROUTINE RMOVE  (KCOLL,JQ,NSET,QSET)
      DIMENSION NSET(1),QSET(1),KCOLL(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
C
C*****THE DUMMY ARRAY KCOLL IS USED AS AN ARGUMENT TO FORCE THE CALL
C*****    BY NAME OPTION ON COMPUTERS SUCH AS THE IBM 360.  THE CALL
C*****    BY NAME OPTION IS REQUIRED FOR PROPER OPERATION OF THIS
C*****    SUBROUTINE.
C
      KCOL = KCOLL(1)
      IF (KCOL) 16,16,2
   16 CALL ERROR (97,NSET,QSET)
      RETURN
C
C*****PUT VALUES OF KCOL IN ATTRIB
C
    2 INDX=(KCOL-1)*IMM
      DO 3 I = 1,IMM
      INDX = INDX + 1
    3 ATRIB(I) = QSET(INDX)
      INDX = (KCOL - 1) * MXX
      DO 10 I =1,IM
      INDX = INDX + 1
   10 JTRIB(I) = NSET(INDX)
      INDX=(KCOL-1)*IMM
      DO 32 I=1,IMM
      INDX = INDX + 1
   32 QSET(INDX) = 0.0
      INDX=(KCOL-1)*MXX
      DO 1300 I= 1,IM
      INDX = INDX + 1
 1300 NSET(INDX) = 0
      INDX=KCOL*MXX
      JL = NSET(INDX - 1)
      JK = NSET(INDX)
      IF (JL-7777) 33,34,33
   33 IF (JK-9999) 35,36,35
   35 INDX = JK * MXX - 1
      NSET(INDX) = JL
      INDX = JL * MXX
      NSET(INDX) = JK
   37 INDX=KCOL*MXX-1
      NSET(INDX) = MFA
      NSET(INDX+1) = 9999
      IF(MFA-8888)234,235,235
  234 INDX=MFA*MXX
      NSET(INDX)=KCOL
  235 MFA=KCOL
      NQ(JQ) = NQ(JQ)-1
      RETURN
   36 INDX = JL * MXX
```

```
      NSET(INDX) = 9999
      MFE(JQ) = JL
      GO TO 37
   34 IF (JK-9999) 38,39,38
   38 INDX = JK *MXX - 1
      NSET(INDX) = 7777
      MLE(JQ) = JK
      GO TO 37
   39 MFE(JQ) = 0
      MLE(JQ) = 0
      GO TO 37
      END
```

```
      FUNCTION RNORM (J)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      RA=DRAND(ISEED)
      RB=DRAND(ISEED)
      V=(-2.0*ALOG(RA))**0.5*COS (6.283*RB)
      RNORM  = V*PARAM (J,4) + PARAM (J,1)
      IF (RNORM -PARAM (J,2)) 6,7,8
    6 RNORM  = PARAM (J,2)
    7 RETURN
    8 IF (RNORM -PARAM (J,3)) 7,7,9
    9 RNORM  = PARAM (J,3)
      RETURN
      END
      SUBROUTINE SAMPL(DEV,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      JP=JTRIB(2)
      JD=JTRIB(3)
      GO TO (1,2,3,4,5,6,7,8,7),JD
    1 DEV=PARAM(JP,1)
      RETURN
    2 DEV=RNORM(JP)
      RETURN
    3 A=PARAM(JP,2)
      B=PARAM(JP,3)
      DEV=UNFRM(A,B)
      RETURN
    4 DEV=ERLNG(JP)
      RETURN
    5 DEV=RLOGN(JP)
      RETURN
    6 CALL NPOSN(JP,NPSSN)
      DEV=NPSSN
      RETURN
    7 DEV=BETA(JP)
      RETURN
    8 DEV=GAMMA5( PARAM(JP,1),PARAM(JP,2),PARAM(JP,3),PARAM(JP,4),NPRNT,
     1JP,ISEED)
      RETURN
      END
      SUBROUTINE SCHAT(NODE,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMON NDT,NDTL,NDTU,NSKD,NSTNO,XSTUS(99)
      COMMON NQN,NQL,NQU,NSTUS(100),SSUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEC(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
C
C*****NEXT IS LOCATION OF FIRST ENTRY IN FILE OF ACTIVITIES WITH START
C*****NODE- NODE. NT IS THE NODE TYPE.
C
      III=0
      NT=NTYPE(NODE)
      NEXT=MFE(NODE)
      IF(NEXT) 4,4,6
    6 GO TO (1,5,1,5),NT
C
C*****GENERATE A RANDOM NUMBER RNUM.
C
    5 RNUM=DRAND(ISEED)
C
C*****OBTAIN STARTING LOCATION IN NSET OF ENTRY.
C
    1 INDXN=(NEXT-1)*MXX
C
C*****TEST IF DETERMINISTIC OR PROBABILISTIC NODE.
C
      GO TO (100,200,100,200),NT
  100 DO 10 I=1,IM
      INDX=INDXN+I
   10 JTRIB(I)=NSET(INDX)
C
C*****OBTAIN SAMPLE FOR ACTIVITY.
C
      CALL SAMPL(DEV,NSET,QSET)
      ATRIB(1)=TNOW+DEV
C
C*****FILE END OF ACTIVITY EVENT IN EVENT FILE.
      CALL FILEM(1,NSET,QSET)
      IF(NRUNS.LT.0)RETURN
      III=1
      GO TO (110,4,110,4),NT
C
C*****DETERMINE IF OTHER ACTIVITIES ARE IN FILE.
C
  110 INDX=INDXN+MX
      NEXT=NSET(INDX)
      IF(NEXT-7777)1,4,23
   23 CALL ERROR(23,NSET,QSET)
      RETURN
    4 IF(III)2,3,2
    3 WRITE(NPRNT,9) TNOW,NODE
    9 FORMAT(/12X,20HSCHAT CALLED AT TIME,F8.2,42H ACTIVITY COULD NOT BE
     1 SCHEDULED FROM NODE,I5/)
      GO TO 23
  200 INDX=(NEXT-1)*IMM+1
C
C*****TEST RNUM AGAINST PROBABILITY (CUM.) OF REALIZING THE ACTIVITY.
C
      IF(QSET(INDX)-RNUM)110,110,100
    2 RETURN
      END
```

```
      FUNCTION UNFRM (A,B)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      RNUM=DRAND(ISEED)
      UNFRM  = A+(B-A)*RNUM
      RETURN
      END
      SUBROUTINE SUMRY (NSET,QSET)
      DIMENSION NSET(1),QSET(1),LIST(7)
      COMMON ID,IM,MFA,MSTOP,MX,MXC,NCLCT,NHIST,NPRNT,NCRDR,NNM,IMN,NPD,
     1NOQ,NPRMS,NRUN,NRUNS,ISEED,TNOW,MXX,IMM,NYR,SEED,
     2MAXQS,MAXNS,ATRIB(2),JTRIB(5),NAME(6),NPROJ,MON,NDAY,JCELS(99,32)
      COMMON MFE(175),MLE(175),NQ(175),PARAM(100,4),SUMA(99,5)
      COMMON NSINK(100),NSKST,NSRC,XLOW(99),NRELP(175),NREL(175),NN,
     1NSKS,NSKSR,NSORC(1),NTYPE(175),WIDTH(99),TOTIM,NCTS,KOUNT(5),
     2NPO(60),NFTBU(175),NABA(270),NREL2(175),NSNR(100),JSINK(100)
      COMMOM NDT,NDTL,NDTU,NSKD,NSTND,XSTUS(99)
      COMMON NQN,NQL,NQU,NSTUS(100),SBUS(100),TLC(100),SQUE(100),XBALK(1
     100),MFEQ(100),MLEQ(100),NPTR(325),MFAQ,IGARBAGE,IMNQ(50),IMNN,
     2QATT(325),K1
      COMMON IGATE(50,2),NUMGQ,ITEST(50)
   21 FORMAT(1H1)
      DIMENSION IFF(2),IAA(2),IBB(2),IDI(2),IDD(2)
      EQUIVALENCE(F,IFF),(A,IAA),(B,IBB),(DI,IDI),(D,IDD)
      DATA (IFF=1HF,1H ),(IAA=1HA,1H ),(IBB=1HB,1H ),(IDI=1HI,1H ),
     1(IDD=1HD,1H )
   10 FORMAT(32X,23HGERT SIMULATION PROJECT,I4,4H BY ,6A2/45X,4HDATE,I3,
     11H/,I3,1H/,I5//)
   15 FORMAT(//31X19H**FINAL RESULTS FOR,I5,14H SIMULATIONS**/)
   16 FORMAT(16X4HNODE3X11HPROB./COUNT,5X4HMEAN,4X8HSTD.DEV.,4X4H= OF,4X
     1,4HMIN.,6X4HMAX.,2X9HNODE TYPE/59X4HOBS./)
   18 FORMAT(1H-20X,23HHISTOGRAM FOR NODE NO.  I4,39H, THE LOWER LIMIT O
     IF THE SECOND CELL =  F10.4,//19X,19H  THE CELL WIDTH =  F10.4 )
   19 FORMAT(1H-30X,17H***FREQUENCIES*** //(21X,11I6))
   25 FORMAT(1H+ 51X, 25HEACH ASTERISK REPRESENTS  F9.3,18H OF A REALIZA
     1TION  //8X,4HCELL  10X,1H19X, 1H29X, 1H39X, 1H49X, 1H59X, 1H69X, 1
     2H79X, 1H89X, 1H99X, 1H0   /2X,5HFREQ. 2X,4HNO.  10(10H1234567890))
      XRUNS = NRUN
      WRITE(NPRNT,21)
      WRITE (NPRNT,10) NPROJ,NAME,MON,NDAY,NYR
      WRITE (NPRNT,15) NRUN
      WRITE(NPRNT,16)
      NCTS1=NCTS+1
      DO 6 K=1,NSKS
      JJ=JSINK(K)
      GO TO (161,162,163,64,65),JJ
  161 CODE=F
      GO TO 12
  162 CODE=A
      GO TO 12
  163 CODE=B
      GO TO 12
   64 CODE=DI
      GO TO 12
   65 CODE=D
   12 DO 6 J=1,NCTS1
      I=(K-1)*NCTS1+J
      JJ=J-1
      IF(SUMA(I,3))5,62,61
    5 WRITE (NPRNT,199)
  199 FORMAT(///36X26HERROR EXIT, TYPE 98 ERROR.)
```

```
   62 WRITE(NPRNT,63) NSINK(K),JJ
   63 FORMAT(I18,I13,9X,18HNO VALUES RECORDED)
      GO TO 6
   61 XS=SUMA(I,1)
      XSS=SUMA(I,2)
      XN = SUMA(I,3)
      AVG=XS/XN
      IF(XN*XSS - XS*XS) 200,200,201
  200 STD=0.0
      GO TO 202
  201 STD=(((XN*XSS)-(XS*XS))/(XN*(XN-1.0)))**0.5
  202 IF(J-1) 41,41,42
   41 IF(JSINK(K)-1) 241,241,223
  223 XN=NSNR(K)
  241 PROB=XN/XRUNS
      WRITE(NPRNT,17) NSINK(K),PROB,AVG,STD,SUMA(I,3),SUMA(I,4),SUMA(I,5
     1),CODE
   17 FORMAT(/I18,F9.4,7X,2F10.4,3X,F7.0,2F10.4,4X,A3)
      GO TO 6
   42 WRITE(NPRNT,20) NSINK(K),JJ,AVG,STD,SUMA(I,3),SUMA(I,4),SUMA(I,5)
   20 FORMAT(I18,I13,3X,2F10.4,3X,F7.0,2F10.4)
    6 CONTINUE
      IF(NQN) 13,13,14
   14 WRITE(NPRNT,22)
   22 FORMAT(1H1//43X15H**QUEUE NODES**//12X4HNODE,4X4HMEAN,4X8HSTD.DEV.
     1,5X,4H= OF,5X4HMIN.,6X4HMAX./40X,4HOBS./)
      I=NSKS*(NCTS+1)+1
      DO 30 NO=NQL,NQU
      WRITE(NPRNT,35)
   35 FORMAT(1H )
      DO 30 JO=1,3
      XS=SUMA(I,1)
      XSS=SUMA(I,2)
      XN=SUMA(I,3)
      AVG=XS/XN
      IF(XN*XSS - XS*XS) 31,31,32
   31 STD=0.0
      GO TO 33
   32 STD=(((XN*XSS)-(XS*XS))/(XN*(XN-1.0)))**0.5
   33 GO TO (51,52,53),JO
   51 WRITE(NPRNT,54) NSINK(NO),AVG,STD,SUMA(I,3),SUMA(I,4),SUMA(I,5)
   54 FORMAT(I15,2F10.4,F10.0,2F10.4,2X27HAVERAGE NUMBER IN THE QUEUE)
      GO TO 30
   52 MY = 0
      DO 9000 ME = 1,NUMGQ
      IF(NSINK(NO) .EQ. IGATE(ME,1)) MY = ME
 9000 CONTINUE
C $$$ SINCE PROCESSING TIME FOR GATED QUEUES ARE MEANINGLESS, SKIP
C $$$ THIS LINE IF CUEUE IS GATED.
      IF(MY .GT. 0) GO TO 30
      WRITE(NPRNT,55) NSINK(NO),AVG,STD,SUMA(I,3),SUMA(I,4),SUMA(I,5)
   55 FORMAT(I15,2F10.4,F10.0,2F10.4,2X30HAVERAGE BUSY TIME OF PROCESSOR
     1)
      GO TO 30
   53 WRITE(NPRNT,56) NSINK(NO),AVG,STD,SUMA(I,3),SUMA(I,4),SUMA(I,5)
   56 FORMAT(I15,2F10.4,F10.0,2F10.4,2X29HAVERAGE BALKERS PER UNIT TIME)
   30 I=I+1
C
C*****HISTOGRAM SECTION
C*****UP TO STATEMENT NO. 86, HEADINGS ARE PRINTED
C*****AND THE DECISION ON WHICH TYPE OF HISTOGRAM TO PRINT
C
   13 DO 82 I=1,NHIST
      IF(WIDTH(I).LT.0.0) WRITE(NPRNT,21)
      XS=ABS(WIDTH(I))
      WRITE(NPRNT,18)NSINK(I),XLOW(I),XS
      IF(WIDTH(I))87,86,86
   86 WRITE(NPRNT,19)(JCELS(I,J),J=1,MXC)
      GO TO 82
C
C*****PICKS THE MAX. CELL SIZE
C
   87 I1=-9999
      DO 83 J=1,MXC
   83 I1=AMAX0(I1,JCELS(I,J))
      IF(I1) 82,82,84
   84 AVE=FLOAT(I1)/100
      WRITE(NPRNT,25)AVE
C
C*****SCALES THE OUPUT FOR THE GRAPHICAL HISTOGRAM
C*****ALSO ENCODES IT AND ACTUALLY PRINTS IT.
C
      DO 85 J=1,MXC
      I2=JCELS(I,J)*100/I1
      IF(I2.GT.0) GO TO 1
      LIST(1)=4H(3X,
      LIST(2)=4HI4,3
      LIST(3)=4HX,I2
      LIST(4)=4H)
      GO TO 85
    1 ENCODE(28,100,LIST)I2
  100 FORMAT(16H(3X,I4,3X,I2,1X,I3,6H(1H*)))
  85  WRITE(NPRNT,LIST) JCELS(I,J),J
   82 CONTINUE
      WRITE(NPRNT,21)
      RETURN
      END
```

APPENDIX A

DELAY NODE QUEUE NODE	A simple queue and a delay node may not be directly connected to each other.
DELAY NODE QUEUE NODE	A delay node cannot be a node to open a gated-queue.
QUEUE COUNT-TYPE	A branch on the input or output side of a queue node may not have a count type, nor may the branch on the input side of a signal node for gated-queue.
QUEUE	When initial number in queue is greater than 0 an extra entity is scheduled to leave the queue in order to make the processor busy.
MARK	The number of realizations for a mark node must be 1. If specified as greater than 1 it will be re-initialized within the program.
QUEUE BRANCH	Not more than one branch may emanate from a deterministic queue node.
TIME PARAM.	Some time distributions (lognormal, beta, gamma, and beta with PERT) reinitialize the time parameter. These time parameters should not be used for more than branch.
MARK NODE I STATISTIC	A mark node may not be placed between another mark node and an interval statistic node.
DELAY NODE COUNT-TYPE	A branch incident to a delay node cannot have a count type.
MARK NODE	Each entity which leaves a mark node is tagged with the time of realization. If the entity reaches a packing node, only the last time is used to tag the outgoing entity.
BALK NODE	A balk node cannot be a statistics node.
SEED	Some numbers are better than others for random number seed.
SINK NODES	Statistics for sink nodes can only be type A or F.

SOURCE NODE	Only one source node is allowed. It may have multiple branches emanating from it.
NETWORK MODIFICATION	A maximum of 60 modifications may be used.
COUNTER TYPES	Maximum number of counter types is 4.
TIME PARAM.	Maximum number of time parameters is 100.
NODES	Maximum number of nodes is 175.
SIMULTANEOUS ACTIVITIES	Maximum number of simultaneous activities which can be scheduled is 325.
STAT. NODE QUEUE NODE COUNT TYPES	(number of stat nodes)x(number of count types + 1) + 3x(number of queue nodes) must be $\leq$ to 99.

APPENDIX B

ERROR MESSAGES

Error Code	Subroutine	
-1	DATAN	A negative node number has been input on the sixth data card type. Correct the card.
-2	DATAN	Number of histograms is less than zero. There maybe something wrong in the number of statistic nodes, check data card type two.
-3	DATAN	Error in network modification input. Number of nodes that can be switched has been exceeded. Action to be taken: reduce number of nodes that are to be switched.
21	GASP	Message is given, ERROR FOR QUEUE NODE***AT TIME***ON RUN. Can't find the node in the list of queue nodes. Error might be that gated queue is not defined as a queue or a delay node is adjacent to a queue node.
22	GASP	More sink nodes have been realized than have been specified. This error should not occur.
23	GASP	Queue node activity is incorrectly scheduled. Too many entities may initially be a queue node. See Appendix A.
23	SCHAT	Error in pointing system for file.
24	GASP	Message is given, ERROR FOR DELAY NODE***AT TIME***ON RUN ***. Delay node number in the list of statistic nodes is missing. A queue node maybe too close to a delay node. See queue node section of users' manual.
25	GASP	An activity cannot be scheduled from the last node that was realized and no other activities are scheduled. The node may not be defined.

Error Code	Subroutine	
26	GASP	After a run has been completed, a clock indicates that there could have been no more activities scheduled. Might indicate an error in system design. Same error as 89.
27	GASP	Queue node activity is incorrectly scheduled. May have too many entities in the queue nodes. See Appendix A.
70	DATAN	Will not initially abort program.
87	FILEM	A printed message saying "OVERLAP SET GIVEN BELOW." There are too many activities and branches to be fitted into the area specified. Action to be taken: enlarge number of activities and branches which can happen at one time in data card type one, field eight, and/or reduce system. See Appendix A.
88	FILEM	Nothing more can be scheduled. Pointer for last activity is less than zero.
89	NFIND	Pointer to next activity to be scheduled is less than zero. Cannot schedule anymore activities.
90	COLCT	An index for collection of data is less than one. Program is at fault.
91(a)	SCHAT	Message is given, SCHAT CALLED AT TIME****ACTIVITY COULD NOT BE SCHEDULED FROM NODE***. Action to be taken: change system design.
(b)	SCHAT	Too many activities waiting to be scheduled. User may have some delay times which are too long in comparision with other times so that too long a list of scheduled events occurs.
97	REMOVE	No more activities can be scheduled. Premature stopage of run. Same as error 89.

Misc. Error Codes	Subroutine	
NO. EXCEEDING 50 OF TSD NODES	DATAN	Put no.'s in field ten of card3; only 50 nodes may be plotted. type three until user has a maximum of 50 activity profile nodes to be plotted.
NODE*****NEEDS A STATISTICS CODE	DATAN	Proper code needed in column 27 of third data type. See sec. 2.1.3. Automatically given first time statistics by default.
NODE*****CAUSES NO. OF STAT/ SINK NODES TO EXCEED THE NO. SPECIFIED	DATAN	Increase number of statistics nodes in card type two, field five. See Appendix A for size limitation.
NODE*****HAS NO HISTOGRAM VALUES	DATAN	Cell width for histogram has not been specified. Automatically a cell width of one is given.
NODE*****HAS OUTPUT BRANCHES WHOSE PROBABILITIES DO NOT SUM TO ONE	DATAN	Check the probabilities of the branches given in data type six.
STANDARD fix taken; K WAS SET = 1.0 FOR BETA****.	BETA	Error in BETA distribution input either mean or standard deviation on time parameter**** is a non-positive number.
STANDARD FIX TAKEN: PARAM ****4 WAS SET TO 1.0 FOR ERLANG	ERLANG	Error in erlang function. Factor is less than 1.0. It is set to 1.0.
STANDARD FIX TAKEN: ALPHA WAS SET = 1.0 FOR GAMMA****.	GAMMA	Error in gamma function. Mean is a non-positive number for parameter number ****, it is set to one.
STANDARD FIX TAKEN: K WAS SET = 1.0 FOR GAMMA ****/GA	GAMMA	Error in function. Standard deviation is a non-positive number for parameter ****, it is set to one.
ERROR IN HISTOGRAM ****HISTO	HISTO	Program tries to collect data for more nodes than specified. Program is stopped at this point.
STANDARD FIX TAKEN:= K WAS SET = 1.0 PGAMMA ****	PGAMMA	Error is calculation of gamma variant. Index of time parameter is less than zero. Check data card type for five.

APPENDIX C

The Motivation for the Development of the Gated Queue

The development of the gated queue **was** necessitated by the need for constructing modules. Modules or subnetworks are used as alternative elements in a network to:

- describe frequently recurring logic operations of moderate complexity,
- correspond to concepts governing the flow of entities which are, in a fashion, far more convenient to the user than primitives, and
- form a set permitting combinations which describe systems or processes on a project level which, in practice, is too complex to be modelled by primitives.

The library of modules (Table C) has been developed with all primitives utilized in GERTS GQ. The development of the modular network modelling approach, commonly referred to as the HAD modular networks(7), is motivated by the need for a clearly defined interface between the practicing engineer and computer readable simulation models. The preparatory effort on both sides of the programmer/user interface is the most time-consuming phase in simulation. Significant improvement results from user-oriented simulation languages which aim to reduce preparation time and costs.

The method goes one step beyond the hierarchical level of the simulation languages, to the level of the sub-unit or module. A module represents an interchangeable modelling unit structured from elementary logic processing components here referred to as "primitives" (8,9,10,11).

Three modules strongly dependent on the gated queue are the DIST, ASSR and UNPK modules. Fig. 1C, 2C and 3C depict the nodal diagram of those modules respectively.

C1. The DIST Module

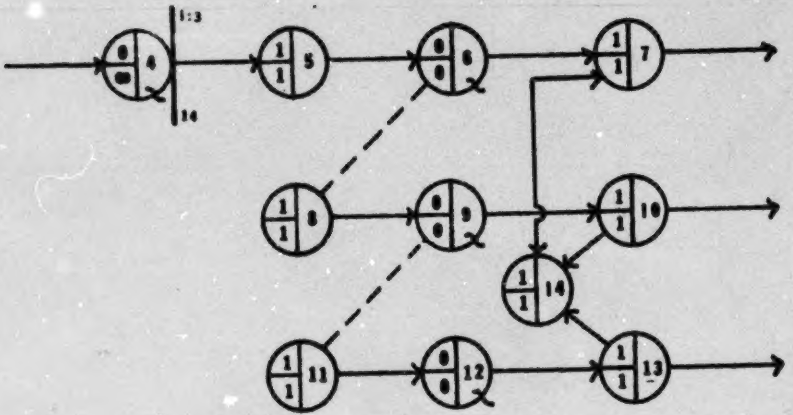
The DISTributor module is used to route an incoming entity to the first idle server encountered in a fixed sequential search of the processor being supplied by the module.

The micro-structure for the DIST module shows three hypothetical processors--P1, P2, P3--supplied by the module which lie outside it. Typically, a processor such as P1 exhibits two states:

- either busy and thus not available to an incoming entity seeking service, or
- idle and ready to accept an arriving entity.

To illustrate the function of the DIST module, we shall follow the flow of a series of entities. An entity enters through the gated-queue, node 4. Then the pulse passes through node 6, another gated-queue, and the gate closes after the pulse has entered the processor. The queue is gated until the switch is turned on later. If a second pulse comes, it is balked over to node 8 and passes through node 9, where the gate closes. This continues until all the gates are closed. When one of the outgoing gates is opened, the incoming gated-queue is opened and lets out one pulse. Everytime an outgoing queue is opened, the input gate has to be open. A common node which has no affect on the program is set up, node 14. This node is set up so that each time a processor is finished, the input queue is opened.

Figure 1C



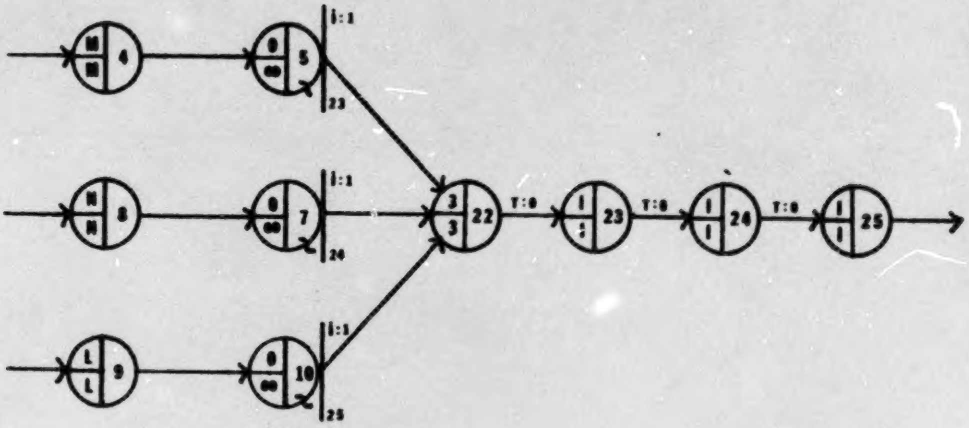
C2. The ASSR Module

In contrast to the PACK module, the ASSR module, code for ASSembleR, has a complex internal structure. The diagram shows the complex task of combining entities from each of the several entry branches according to an exact algorithm or "recipe." In the micro-structure, we have several input stages, each feeding a single PACK module (node 22 used here as a constituent of the larger module) which acts as a flow compactor and output stage for the ASSR.

As many input stages as desired may be specified; here, as a convenient choice, three input stages have been used. Items are packed, then held in the queue until the compactor (node 22) has been realized.

Now that the "recipe" corresponding to the first input stage has been achieved, the remaining input stages maybe similarly realized. Finally, the compactor sub-module will be realized causing the emission of a single output transaction representing a single unit of the "recipe," that is, a single complete assembly.

Figure 2C



C3. The UNPK Module

The UNPacK module emits a predetermined number of entities for each entity (or pulse) which enters it. This feature facilitates modelling of unloading or unpacking situations in construction or industrial processes.

Unlike the IFPG module, the UNPK module is not a "one-shot" device, but automatically resets upon the completion of an emitted pulse train. Comparison of the GERTS micro-structure of this module with the IFPG module is instructive. The two modules are quite similar internally, both contain identical pulse generators and counter sections. The realization of node 8 triggers a network modification to terminate the output pulse train. The two modules differs only in the details of the entry section of the module.

The IFPG module permanently blocks the entry port to the module once the generator has been initiated. In the UNPK module, a gated-queue holds the pulse while the generator is "busy", then lets out the entities as the generator becomes available.

Figure 3C

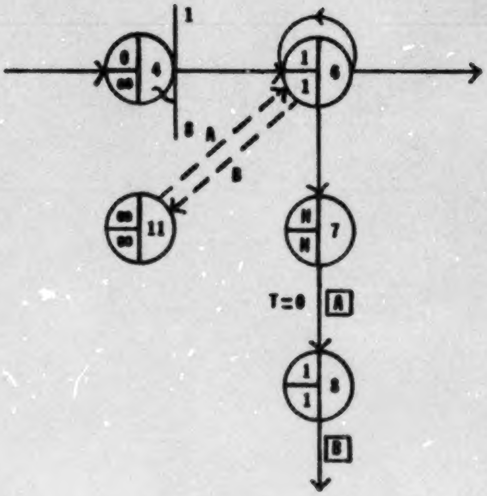


Table C

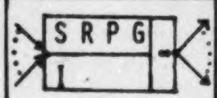
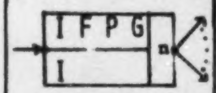
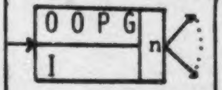
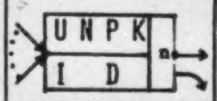
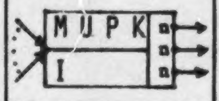
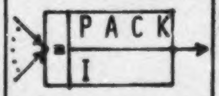
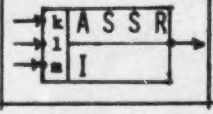
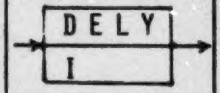
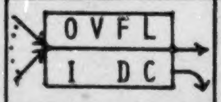
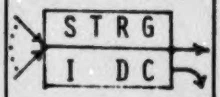
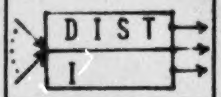
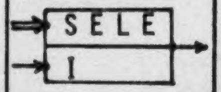
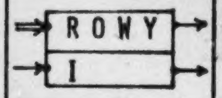
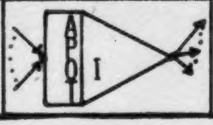
CATEGORY NAME OF MODULE	SYMBOL	I = MODULE ID NUMBER D = DIVERSION DESTINATION C = MAXIMUM CAPACITY n = PROLIFERATE NUMBER k, l, m = CONSOLIDATE NUMBER
PROLIFERATIVE SIMPLE REPETITIVE PULSE GENERATOR		Simple Repetitive Pulse Generator: generates transactions at constant or random time intervals throughout the simulation run.
INITIAL FINITE PULSE GENERATOR		Initial Finite Pulse Train Generator: creates a sequence, consisting of a pre-assigned finite number of transactions, for entering into the system at random or at constant time intervals.
On and Off Pulse Generator		On and Off Pulse Generator: generates a sequence of transactions into a system and can be set and reset by pre-assigned activities.
UNPACK		UNPacK: divides the units of an entity flow into a larger number of subunits. A single flow unit arriving at this module generates a pre-assigned number of flow units at its output.
MULTIPLE UNPACK		The Multiple UnPacK: combines the features of the UNPK and the DIST modules. A single entity enters the module and searches for a server which is available to unpack it.
CONSOLIDATIVE PACK		PACK: consolidates transactions; entering transactions can accumulate in the module until a specified capacity N is attained.
ASSEMBLER		ASSembleR: combines a pre-defined set of entities for future processing. The output activity cannot be scheduled until at least one unit has entered along each input branch.

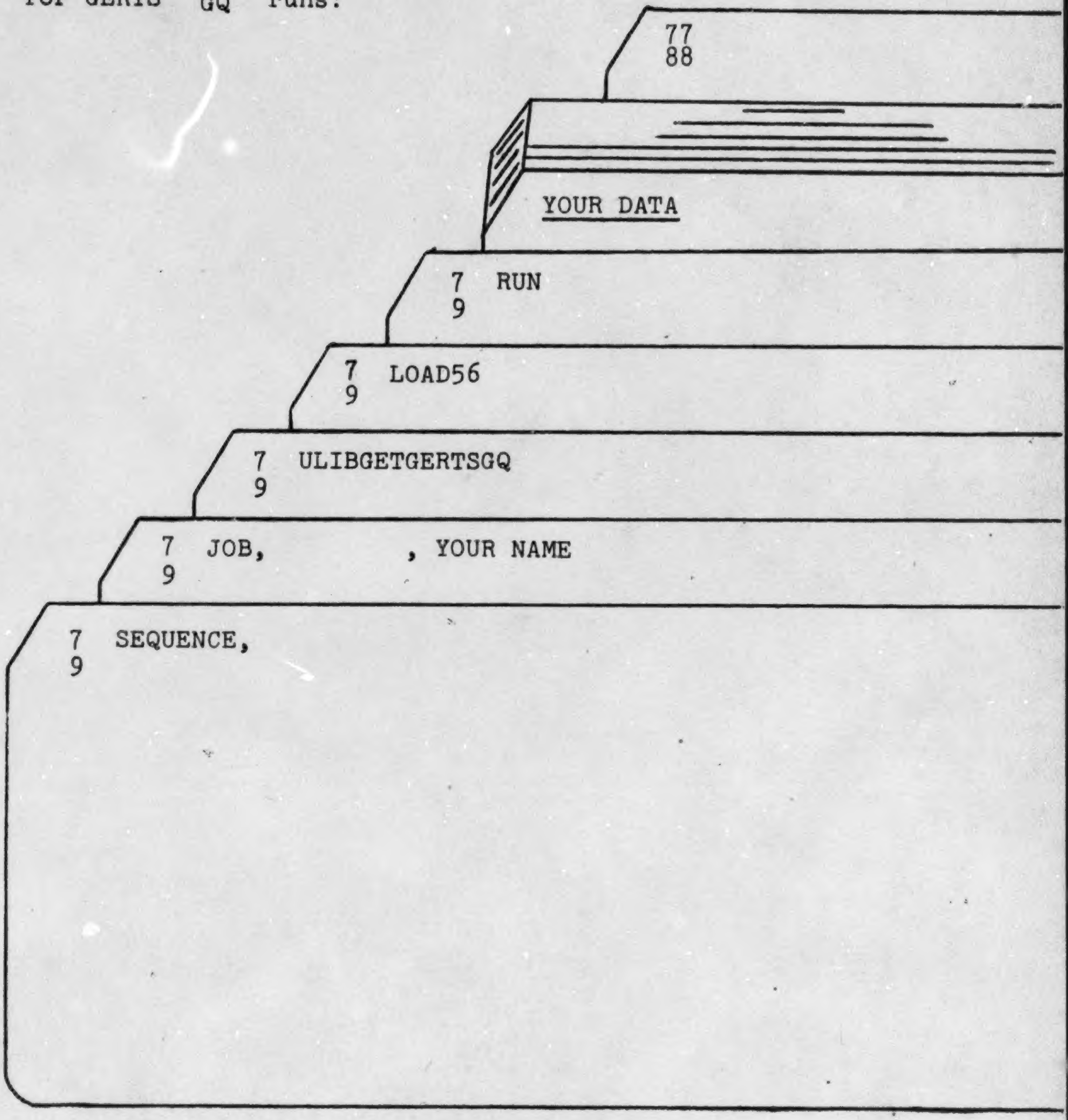
Table C contiued

CATEGORY NAME OF MODULE	SYMBOL	I = MODULE ID NUMBER n = PROLIFERATE NUMBER D = DIVERSION DESTINATION k, l, m = CONSOLIDATE NUMBER C = MAXIMUM CAPACITY
FLOW-THROUGH DELAY		DELaY: delays or holds up transactions at a particular point in the network.
OVERFLOW		OVerFLow: a queueing module providing for diversion of transactions to alternate locations whenever the queue is full.
STORAGE		SToRaGe: Incoming transactions are accumulated ("stockpiled") and are withdrawn, individually or in groups, only as service facilities ourside the module become available.
CHANNELED DISTRIBUTOR		DISTributor: routes an incoming entity to the first idle server encountered in a sequential search of the processors being supplied by the module.
SELECTOR		SELEctor: preferentially reoutes higher ranking entities to a server while holding back lower ranking ones in a queue.
RIGHT OF WAY		Right Of WaY: preferentially controls entity flow through a traffic intersection with a high (H) and a low (L) priority entry path and with two probabilistically selected exit paths.
APPORTION		APpOrTion: probabilistically directs each entering entity to one of its output branches.

Appendix D

Sample job set up for running GERTS GQ

The following deck structure should be used for GERTS GQ runs:


77
88
YOUR DATA
7 RUN
9
7 LOAD56
9
7 ULIBGETGERTSGQ
9
7 JOB, , YOUR NAME
9
7 SEQUENCE,
9

Bibliography

1. Pritsker, A.A.B., and Happ, W.W., " GERT: Part I-Fundamentals," J. Ind. Eng., Vol. XVII, No. 5, 1966, pp. 267-274.

2. Dabaghian, L., Akiba, Y. and Happ, W. W., "Simulation and Modelling Techniques for "Q" Networks", Proceedings, Seventh Asilomar Conference on Circuits, Systems, and Computers, Monterey, CA, November, 1973.

3. Pritsker, A. A. B. and Burgess, R. R., "The GERT Simulation Programs: GERTS III, GERTS III Q, GERTS III C, and GERTS III R," NASA/ERC Contract NAS-12-2113, Virginia Polytechnic Institute, Department of Industrial Engineering, 1970.

4. Blom,N., Duran,G., "The Gated Queue Node: Properties of a New Element for Stochastic Queueing Networks" Report to NSF, Grant GY-11496, School of Engineering California State University, Sacramento, CA, 1974.

5. Akiba, Y., W.W. Happ, "Activity Profiles: A Validation Technique for Logic Modules in Pulse Networks," Proceedings, Eighth Asilomar Conference on Circuits, Systems, and Computers, Monterey, CA December 1974.

6. Pritsker, A. A. B., and Whitehouse, G. E., "GERT: Part II - Probabilistic and Industrial Engineering Applications," J. Ind. Eng., Vol XVII, No. 6, 1966, pp. 293-301.

7. Akiba, Y., "Decision and Commodity Flow for Cargo Facilities," Proceedings, Greater Los Angeles Area Transportation Symposium, Los Angeles, CA, Oct. 1974.

8. Halpin, D. W. and Happ, W. W., "Digital Simulation of Equipment Allocation for Corps of Engineer Construction Planning", Proceedings, Seventeenth Conference on Design and Experiments in Army Research, Washington, D. C., Oct. 1971.

9. Akiba, Y., Dabaghian, L. and Happ, W. W., "Validation of component and System Performance in Modular Queueing Networks," Proceedings, Seventh Asilomar Conference on Circuits, Systems, and Computers, Monterey, CA, November, 1973.

10. Dabaghian, L., Akiba, Y., and Happ, W. W. "Network Modules to Simulate Quantized Entity Flow," Proceedings, Joint Automatic Control Conf., Austin, Texas, June 19, 1974.

11. Halpin, D. W., "An Investigation of the Use of Simulation Networks for Modelling Construction Operations," Ph. D. Dissertation, Graduate College of the University of Illinois, Urbana, Champaign, 1973.